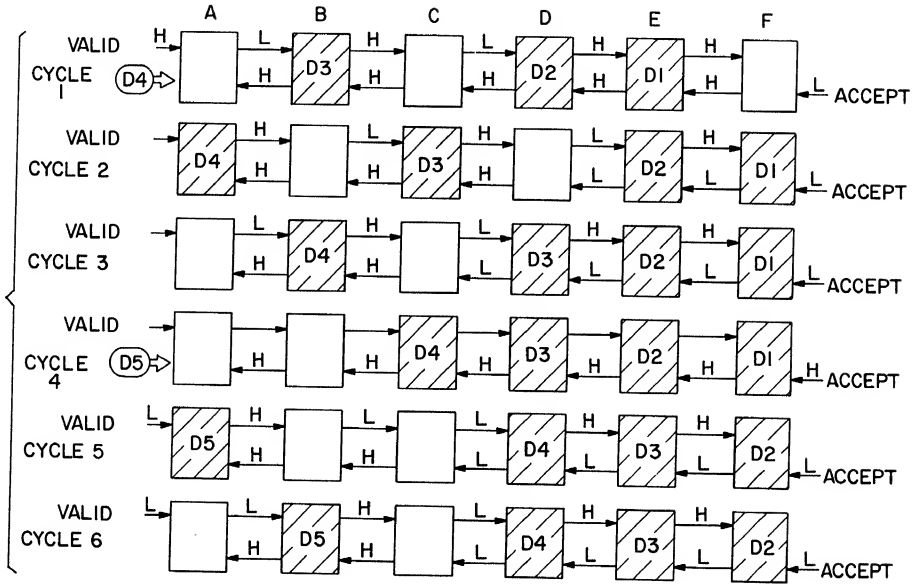

A
B
C
D
E
F
VALID
CYCLE 1
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 2
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 3
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 4
D5
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 5
D5
D4
D3
D2
ACCEPT
VALID
CYCLE 6
D5
D4
D3
D2
ACCEPT
H
L

FIG. 1

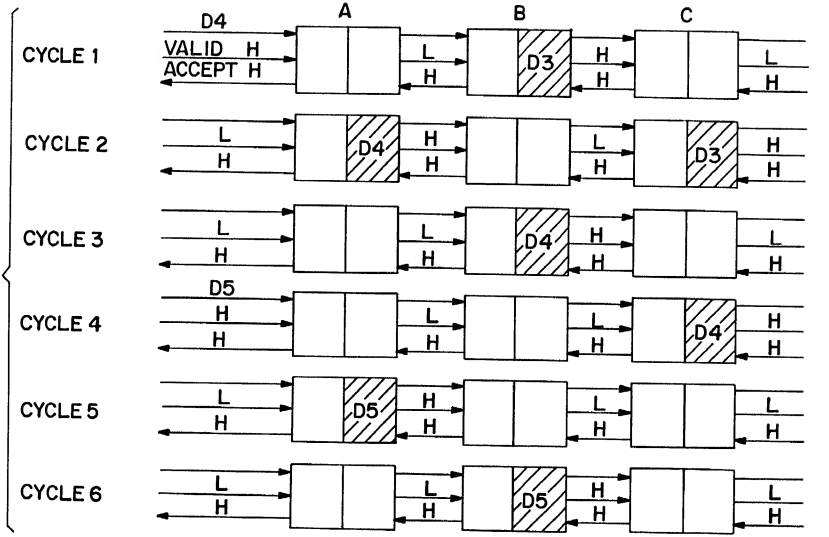

A
B
C
CYCLE 1
CYCLE 2
CYCLE 3
CYCLE 4
CYCLE 5
CYCLE 6
D4
VALID H
ACCEPT H
D3
D4
D3
D4
D5
D4
D5
D5
L
H

FIG. 2(A)

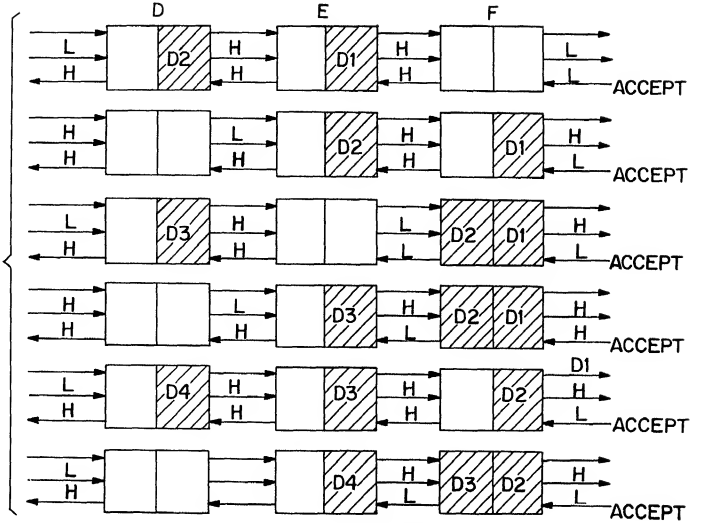


FIG. 2(B)

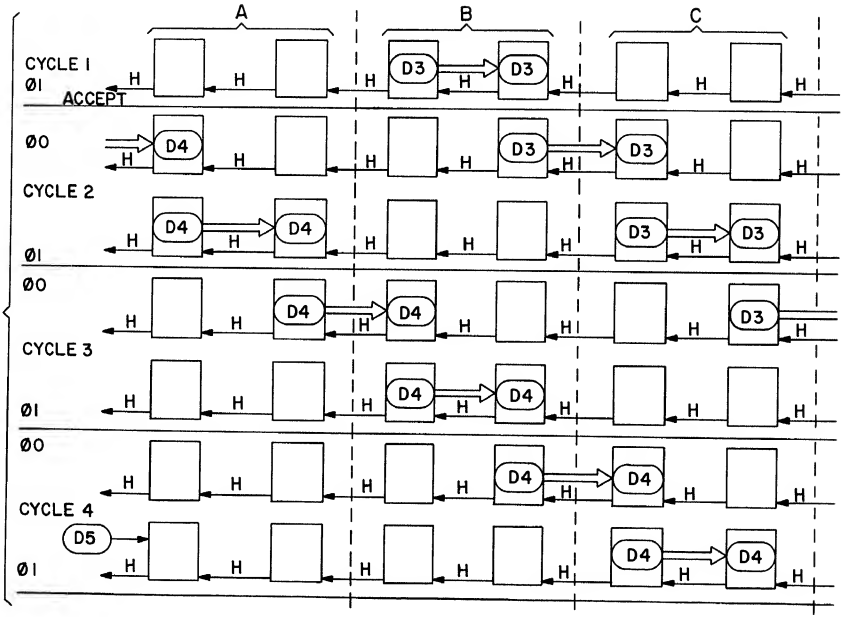


FIG. 3A-1

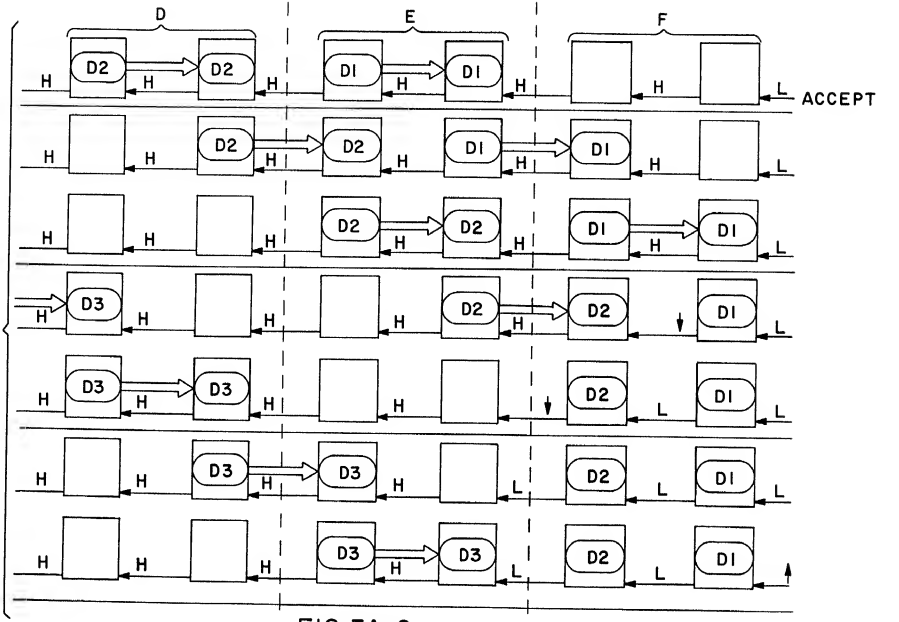


FIG. 3A-2

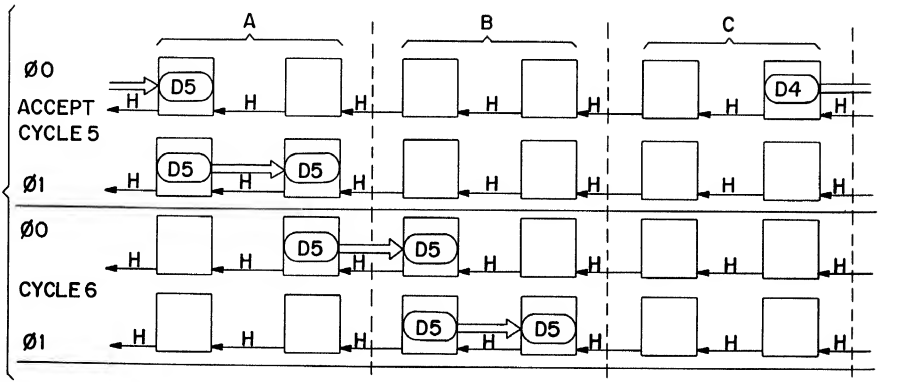


FIG. 3B-1

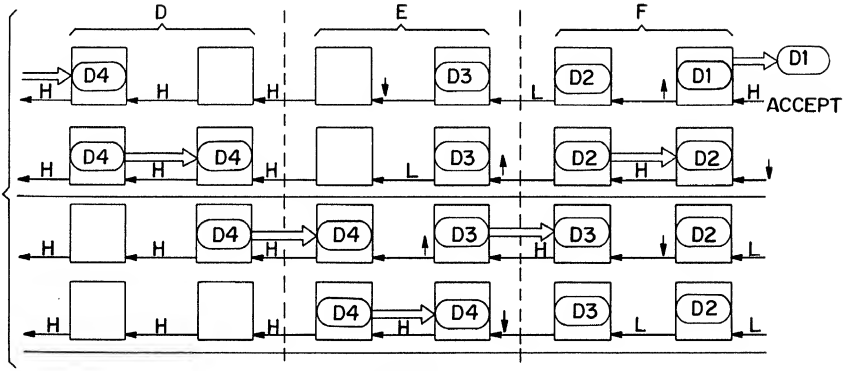


FIG. 3B-2

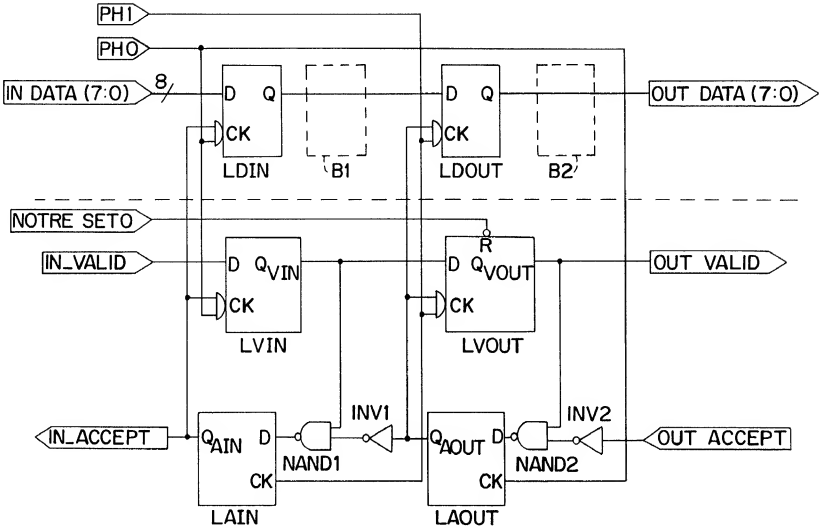


FIG. 4

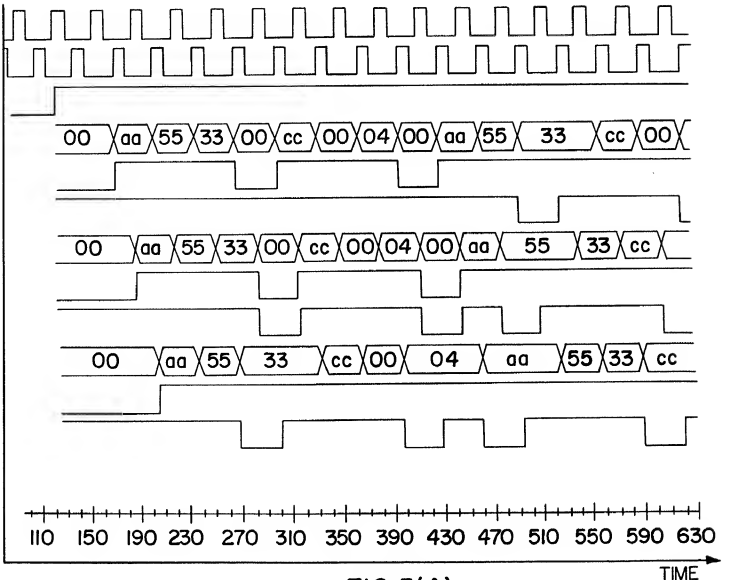


FIG. 5(A)

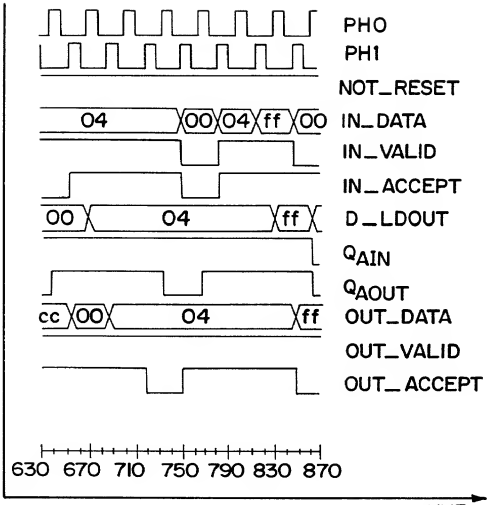

PH0
PH1
NOT_RESET
04
00
04
ff
00
IN_DATA
IN_VALID
IN_ACCEPT
00
04
ff
D_LDOUT
Q_{AIN}
Q_{AOUT}
cc
00
04
ff
OUT_DATA
OUT_VALID
OUT_ACCEPT
630 670 710 750 790 830 870
TIME

FIG. 5(B)

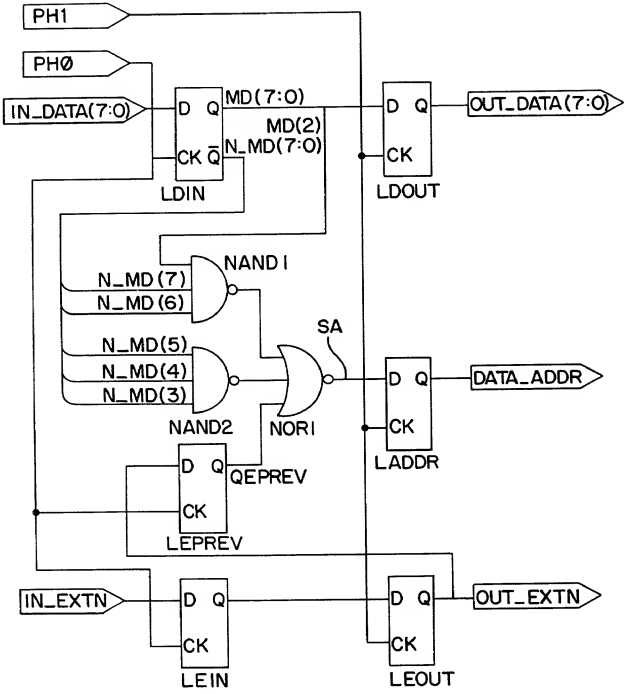

PH1
PHØ
IN_DATA(7:0)
D Q
CK Q̄
LDIN
MD(7:0)
MD(2)
N_MD(7:0)
D Q
CK
LDOUT
OUT_DATA(7:0)
NAND1
N_MD(7)
N_MD(6)
SA
N_MD(5)
N_MD(4)
N_MD(3)
NAND2
NOR1
D Q
CK
LADDR
DATA_ADDR
D Q
QEPREV
CK
LEPREV
IN_EXTN
D Q
CK
LEIN
D Q
CK
LEOUT
OUT_EXTN

FIG. 6

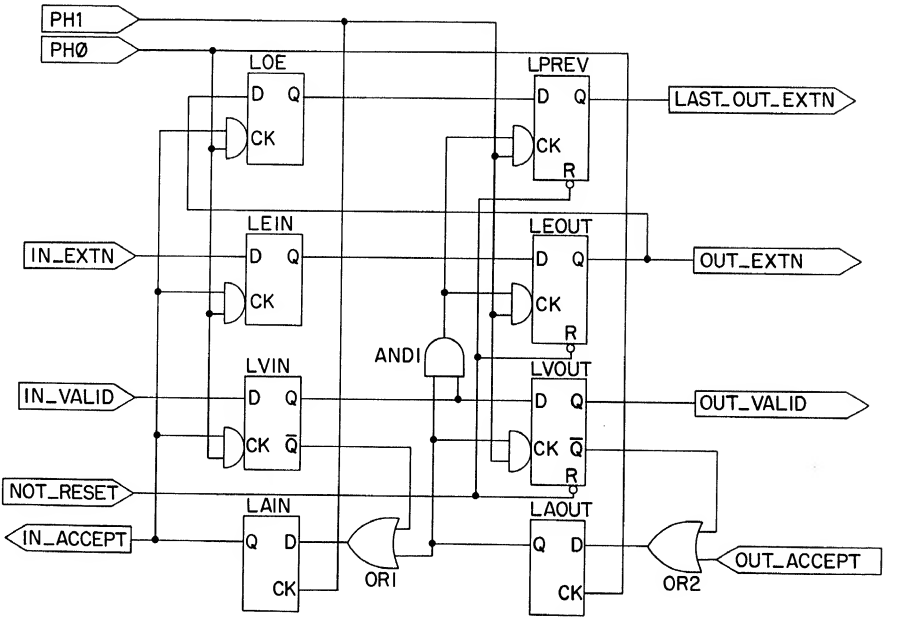


FIG. 7

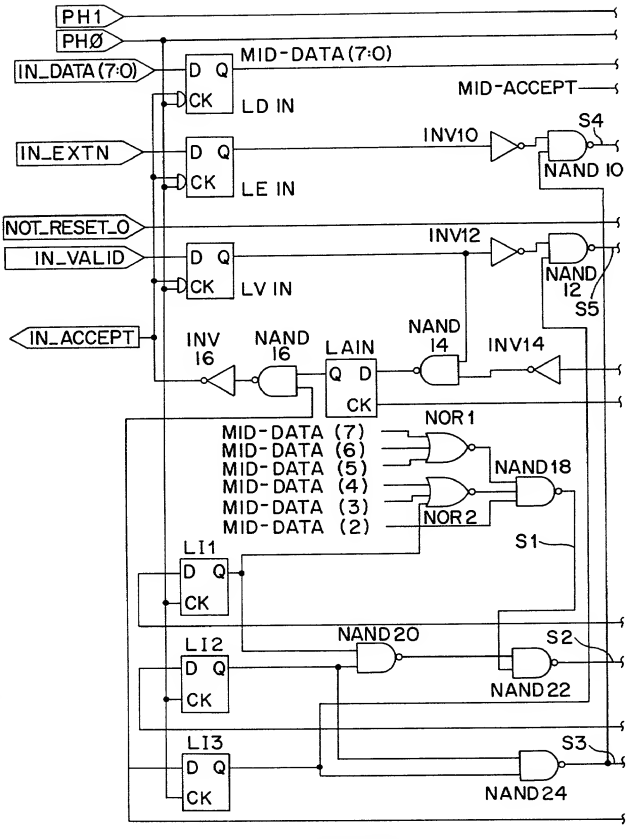

PH1
PHØ
IN_DATA(7:0)
D Q
CK
LD IN
MID-DATA(7:0)
MID-ACCEPT
IN_EXTN
LE IN
INV10
S4
NAND 10
NOT_RESET_O
IN_VALID
LV IN
INV12
NAND 12
S5
IN_ACCEPT
INV 16
NAND 16
LAIN
Q D
CK
NAND 14
INV14
NOR1
MID-DATA (7)
MID-DATA (6)
MID-DATA (5)
MID-DATA (4)
MID-DATA (3)
MID-DATA (2)
NOR2
NAND18
S1
LI1
LI2
LI3
NAND20
S2
NAND22
S3
NAND24

FIG. 8(A)

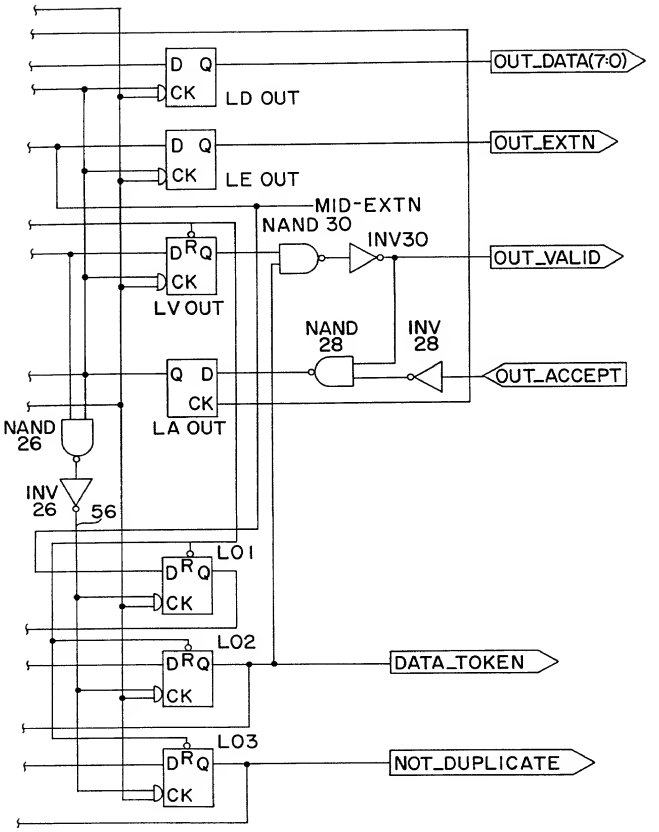

D Q
CK
LD OUT
OUT_DATA(7:0)
D Q
CK
LE OUT
OUT_EXTN
MID-EXTN
NAND 30
INV30
D R Q
CK
LV OUT
OUT_VALID
NAND
28
INV
28
Q D
CK
LA OUT
OUT_ACCEPT
NAND
26
INV
26
56
LO1
D R Q
CK
LO2
D R Q
CK
DATA_TOKEN
LO3
D R Q
CK
NOT_DUPLICATE

FIG. 8(B)

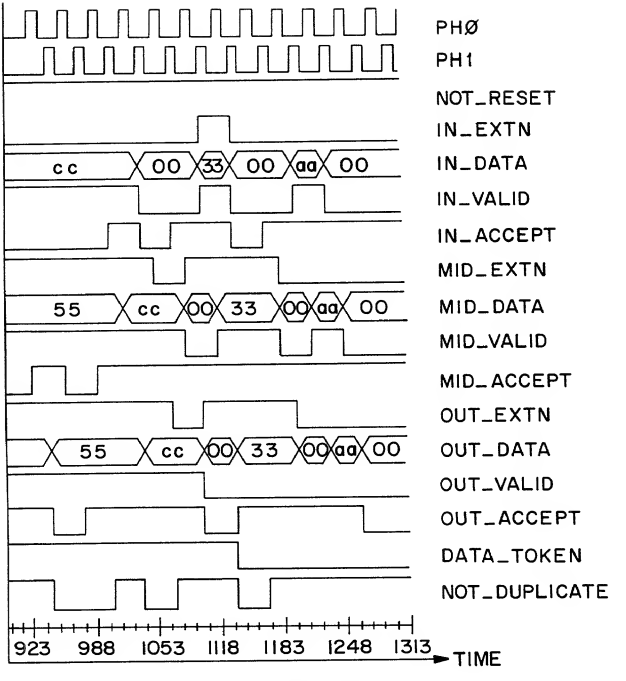

PHØ
PH1
NOT_RESET
IN_EXTN
IN_DATA
IN_VALID
IN_ACCEPT
MID_EXTN
MID_DATA
MID_VALID
MID_ACCEPT
OUT_EXTN
OUT_DATA
OUT_VALID
OUT_ACCEPT
DATA_TOKEN
NOT_DUPLICATE
cc
00
33
00
aa
00
55
cc
00
33
00
aa
00
55
cc
00
33
00
aa
00
923
988
1053
1118
1183
1248
1313
TIME

FIG. 9(B)

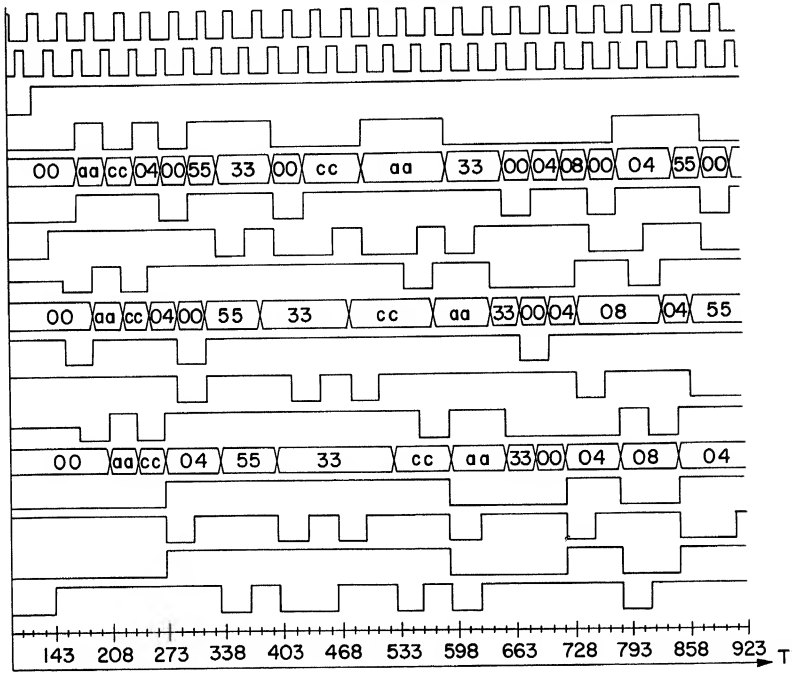

00 aa cc 04 00 55 33 00 cc aa 33 00 04 08 00 04 55 00
00 aa cc 04 00 55 33 cc aa 33 00 04 08 04 55
00 aa cc 04 55 33 cc aa 33 00 04 08 04
143 208 273 338 403 468 533 598 663 728 793 858 923
TIME

FIG. 9(A)

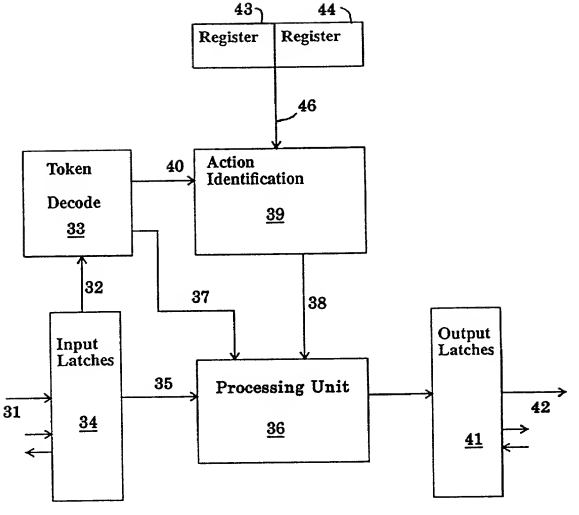

43
44
Register
Register
46
Token
Decode
33
40
Action
Identification
39
32
37
38
Input
Latches
34
35
Processing Unit
36
Output
Latches
41
42
31

FIG. 10

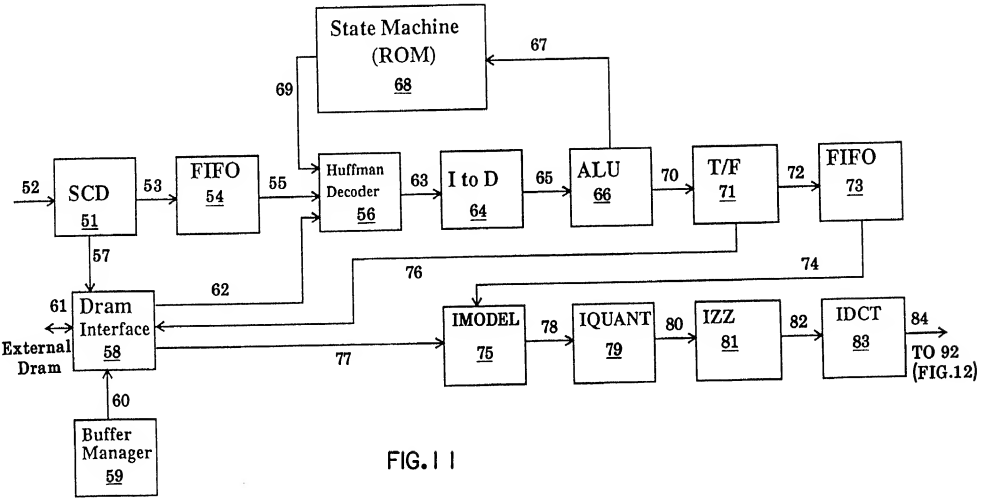


FIG. 11

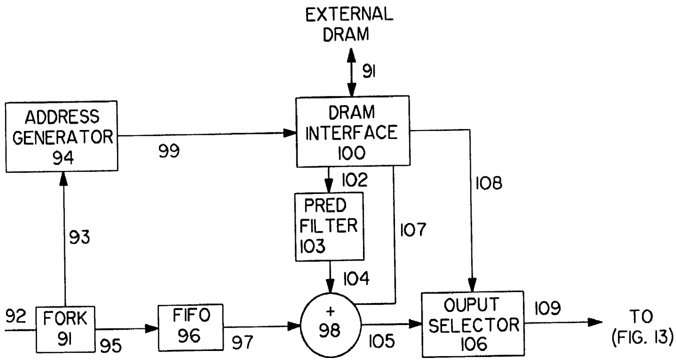

EXTERNAL
DRAM
91
ADDRESS
GENERATOR
94
99
DRAM
INTERFACE
100
102
108
PRED
FILTER
103
107
93
104
92
FORK
91
95
FIFO
96
97
+
98
105
OUPUT
SELECTOR
106
109
TO
(FIG. 13)

FIG. 12

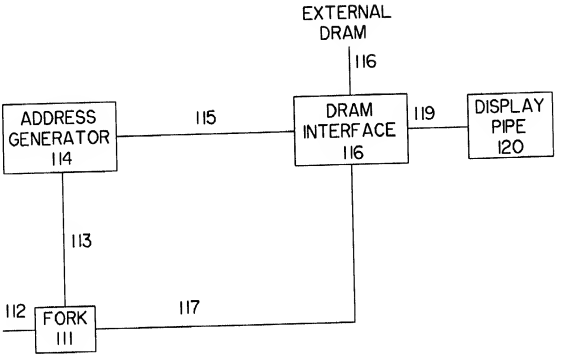


FIG. 13

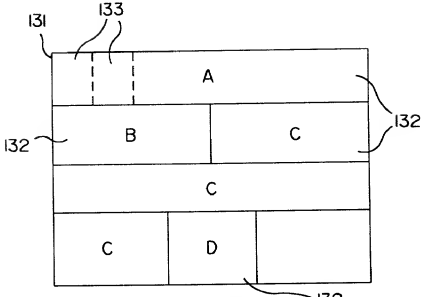


FIG. 14a

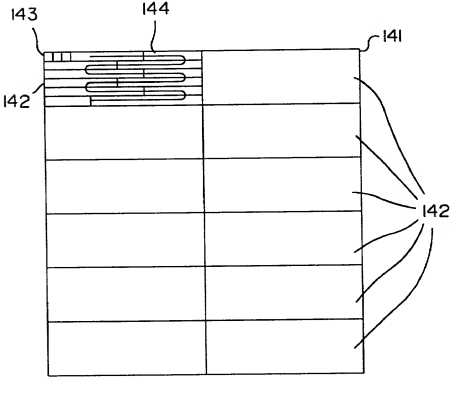


FIG. 14b

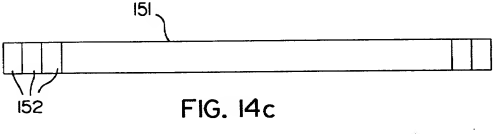


FIG. 14c

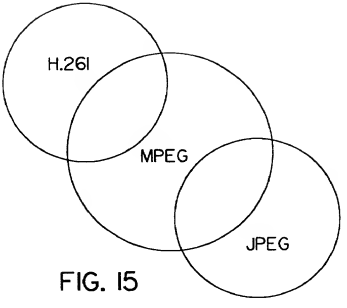

H.261
MPEG
JPEG

FIG. 15

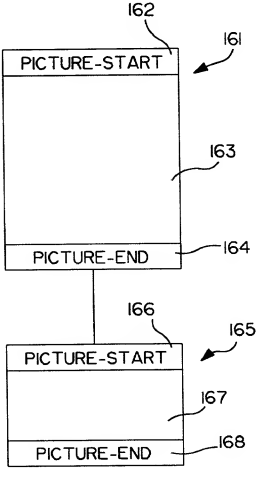

162
161
PICTURE-START
163
PICTURE-END
164
166
165
PICTURE-START
167
PICTURE-END
168

FIG. 16

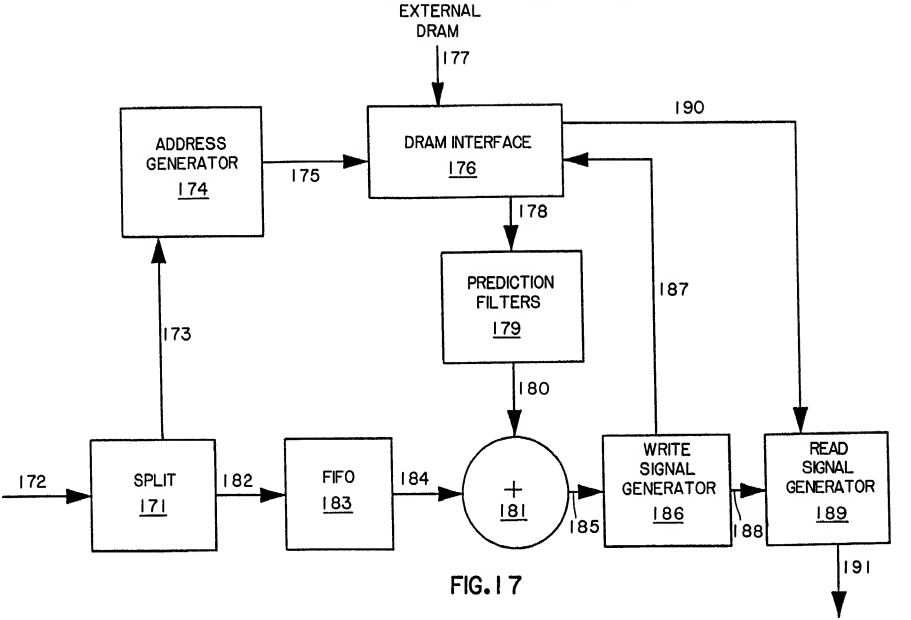


FIG. 17

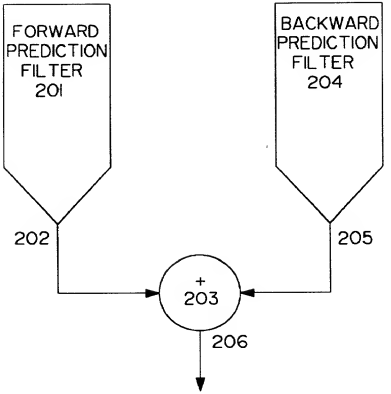

FORWARD
PREDICTION
FILTER
201
BACKWARD
PREDICTION
FILTER
204
202
205
+
203
206

FIG. 18

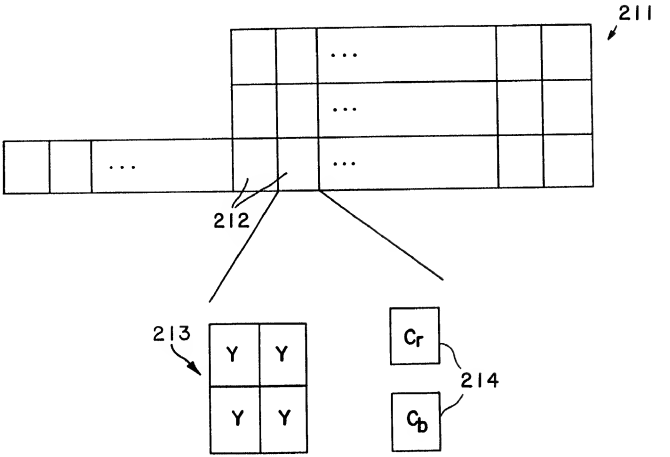


FIG. 19

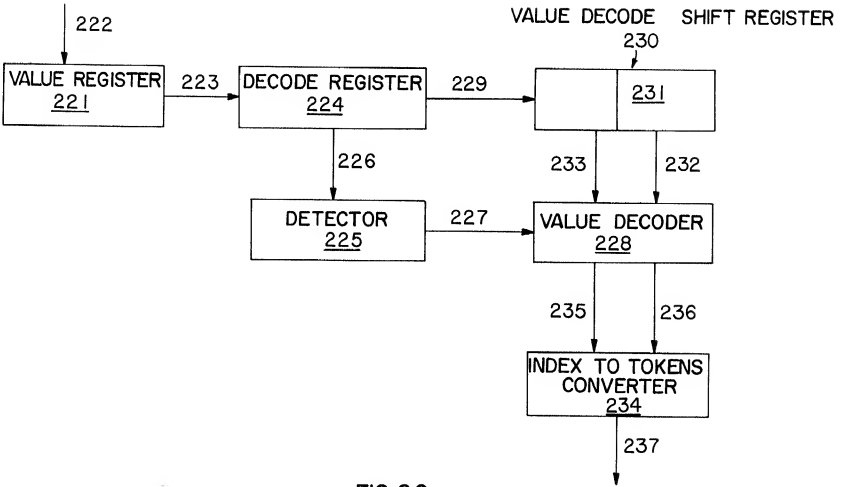


FIG.20

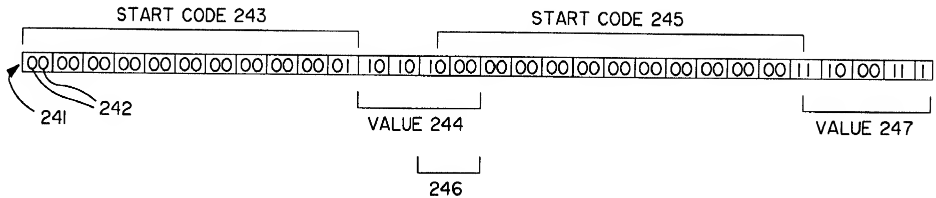

START CODE 243
START CODE 245
00 00 00 00 00 00 00 00 00 00 01 10 10 10 00 00 00 00 00 00 00 00 00 00 00 11 10 00 11 1
241
242
VALUE 244
246
VALUE 247

FIG. 21

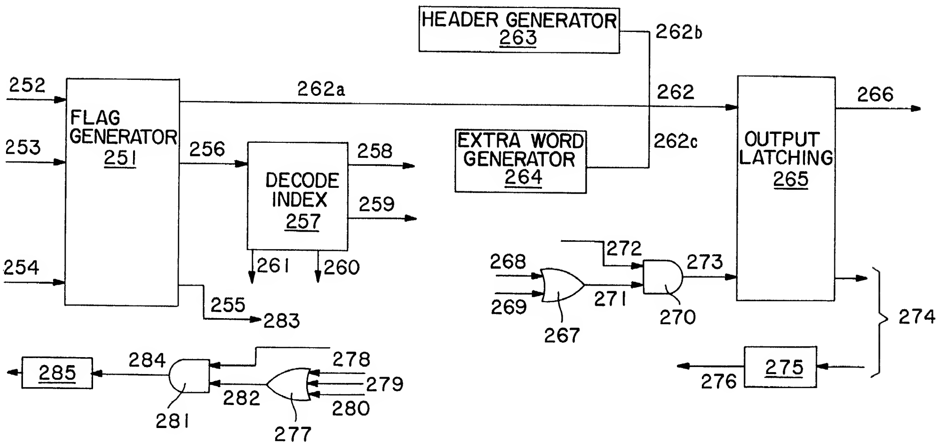

HEADER GENERATOR
263
262b
252
253
254
FLAG GENERATOR
251
262a
262
262c
256
DECODE INDEX
257
258
259
261
260
255
283
EXTRA WORD GENERATOR
264
OUTPUT LATCHING
265
266
272
268
269
267
271
270
273
274
275
276
284
285
281
282
277
278
279
280

FIG.22

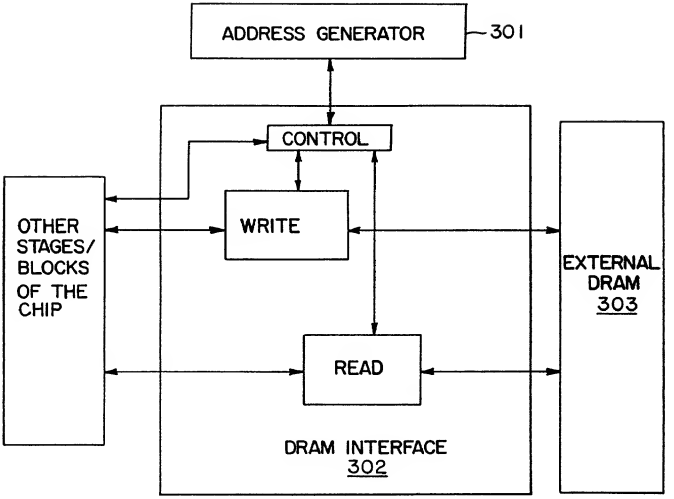

ADDRESS GENERATOR
301
CONTROL
WRITE
READ
OTHER STAGES/ BLOCKS OF THE CHIP
EXTERNAL DRAM 303
DRAM INTERFACE 302

FIG.23

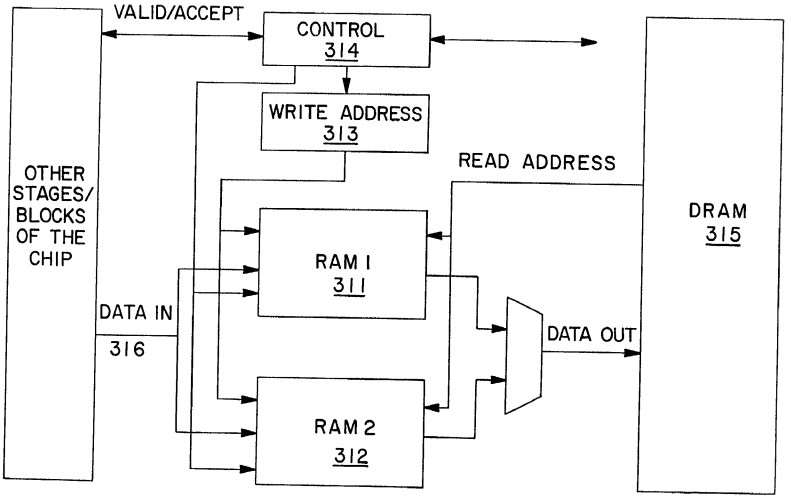


FIG.24

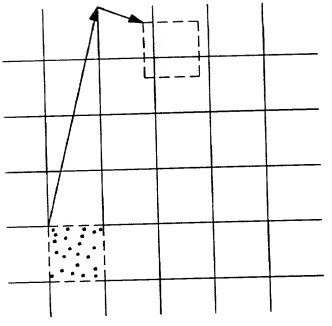


FIG. 25

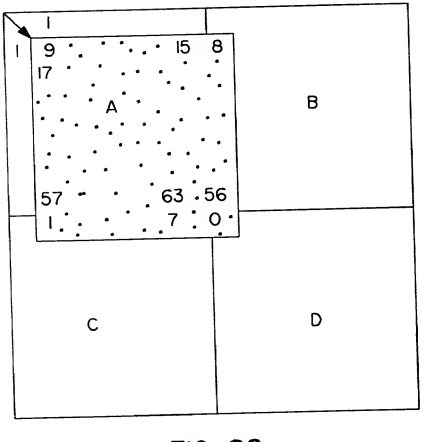


FIG. 26

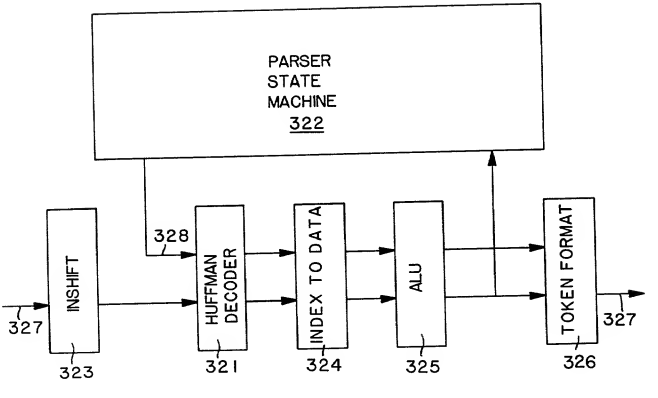

PARSER
STATE
MACHINE
322
328
INSHIFT
HUFFMAN DECODER
INDEX TO DATA
ALU
TOKEN FORMAT
327
327
323
321
324
325
326

FIG.27

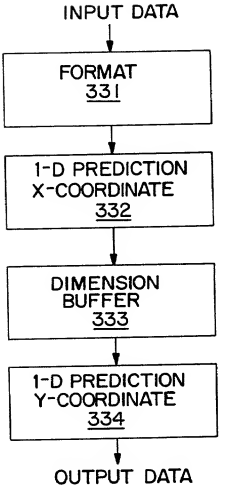

INPUT DATA
FORMAT
331
1-D PREDICTION
X-COORDINATE
332
DIMENSION
BUFFER
333
1-D PREDICTION
Y-COORDINATE
334
OUTPUT DATA

FIG.28

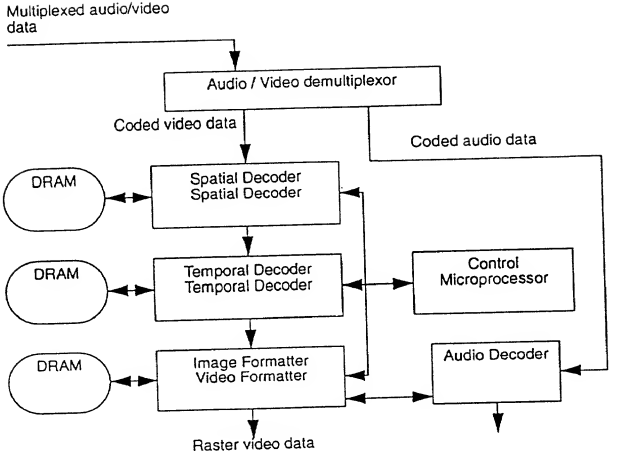

Multiplexed audio/video data
Audio / Video demultiplexor
Coded video data
Coded audio data
DRAM
Spatial Decoder
Spatial Decoder
DRAM
Temporal Decoder
Temporal Decoder
Control
Microprocessor
DRAM
Image Formatter
Video Formatter
Audio Decoder
Raster video data

FIG.29

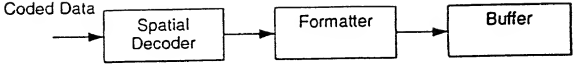


FIG.30

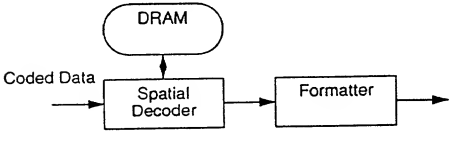


FIG.31

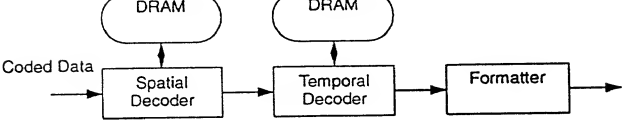


FIG.32

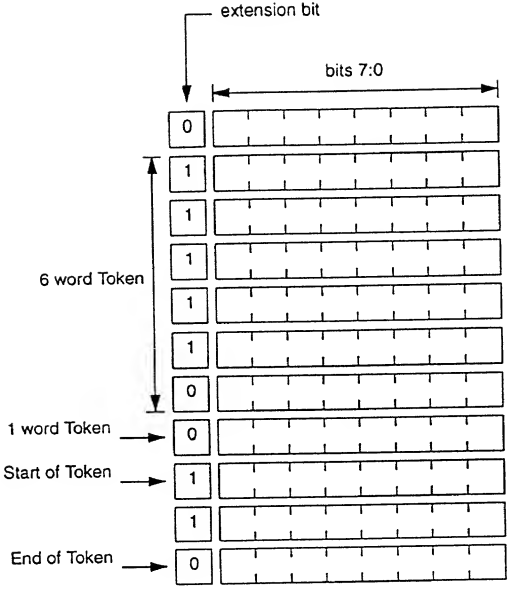

extension bit
bits 7:0
0
1
1
1
1
1
0
6 word Token
1 word Token
0
Start of Token
1
1
End of Token
0

FIG.33

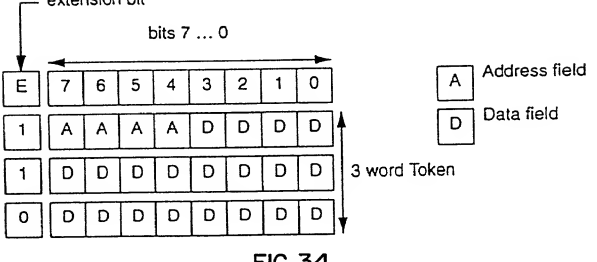

extension bit
bits 7 ... 0
E 7 6 5 4 3 2 1 0
1 A A A A D D D D
1 D D D D D D D D
0 D D D D D D D D
3 word Token
A Address field
D Data field

FIG.34

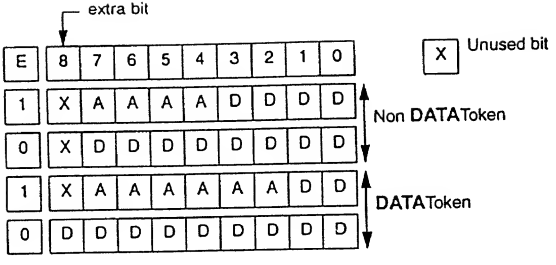
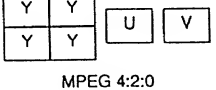
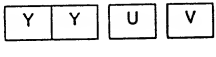


FIG.35



MPEG 4:2:0
macroblock

FIG.36A



JPEG 2:1:1
macroblock

FIG.36B

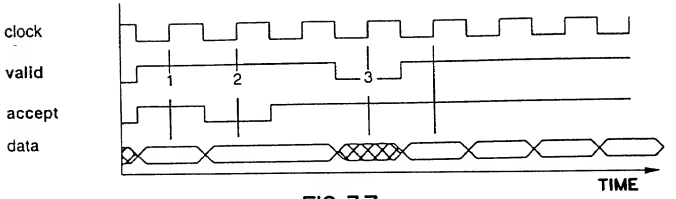

clock
valid
accept
data
1
2
3
TIME

FIG.37

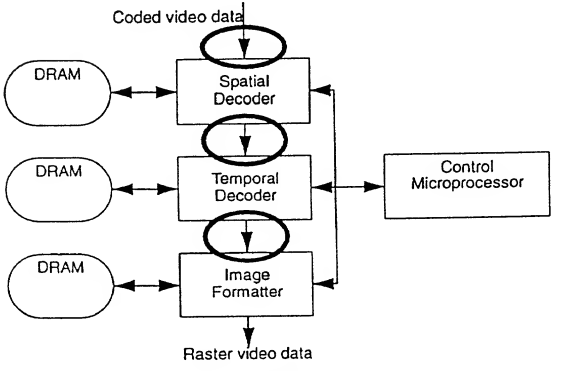

Coded video data
DRAM
Spatial Decoder
DRAM
Temporal Decoder
Control Microprocessor
DRAM
Image Formatter
Raster video data

FIG.38

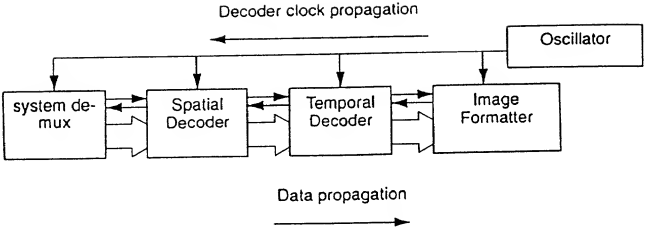

Decoder clock propagation
Oscillator
system de-mux
Spatial Decoder
Temporal Decoder
Image Formatter
Data propagation

FIG.39

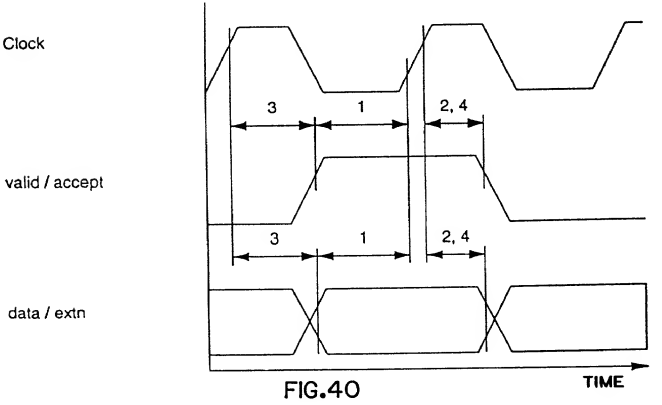

Clock
valid / accept
data / extn
3
1
2, 4
3
1
2, 4
TIME

FIG.40


Access Start
Data Transfer
Default State

FIG.41

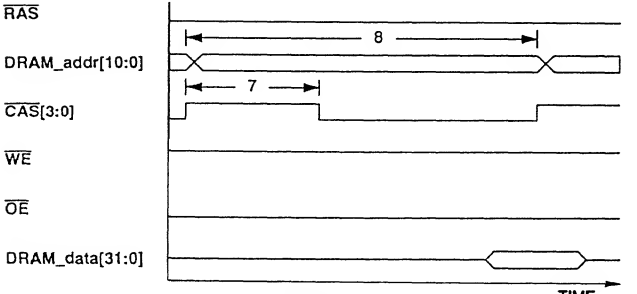

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.42

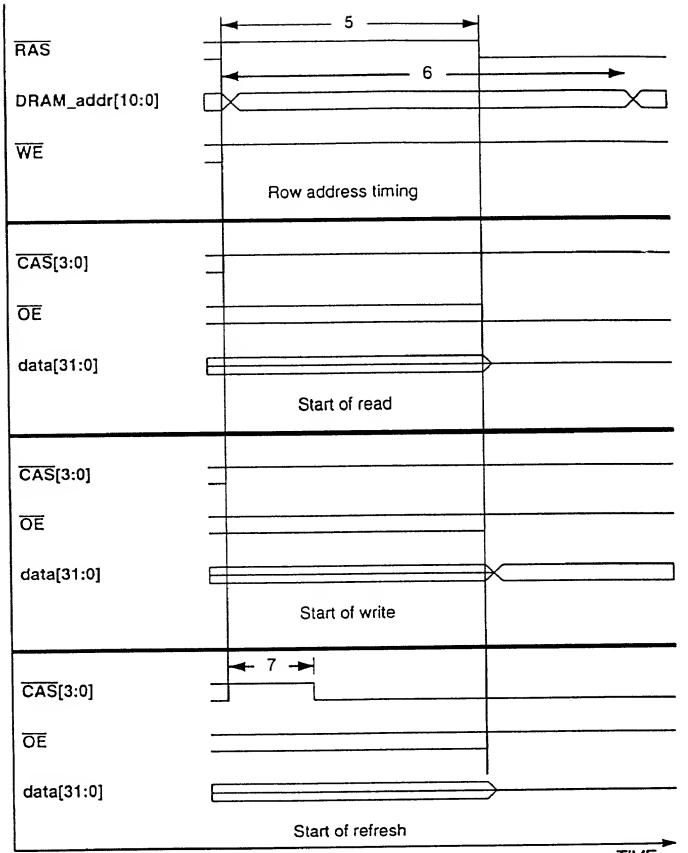

5
RAS
6
DRAM_addr[10:0]
WE
Row address timing
CAS[3:0]
OE
data[31:0]
Start of read
CAS[3:0]
OE
data[31:0]
Start of write
7
CAS[3:0]
OE
data[31:0]
Start of refresh
TIME

FIG.43

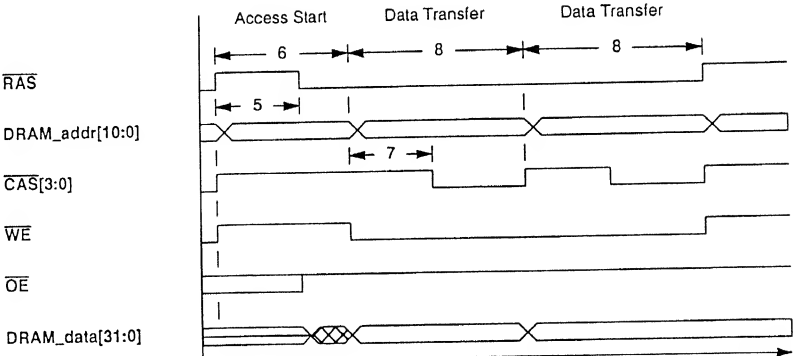


FIG.44

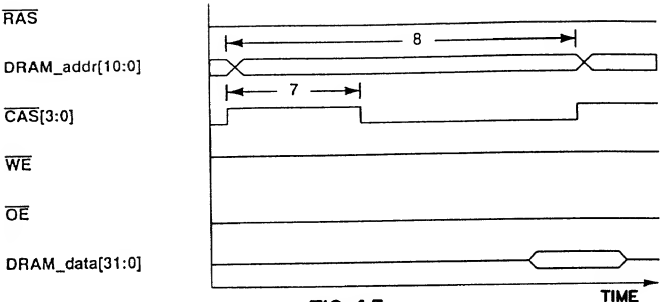

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.45

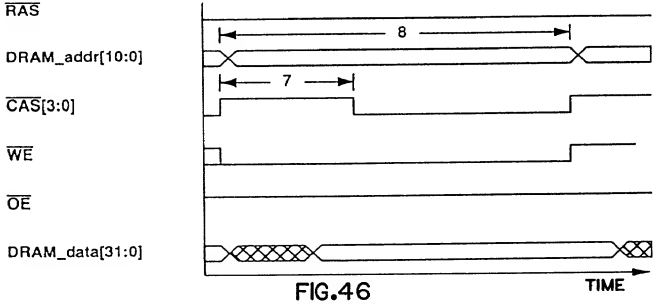

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.46

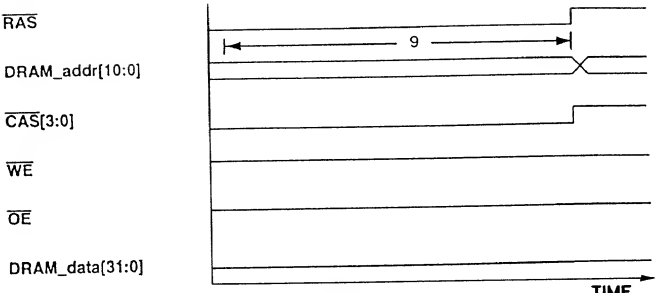

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
9
TIME

FIG.47

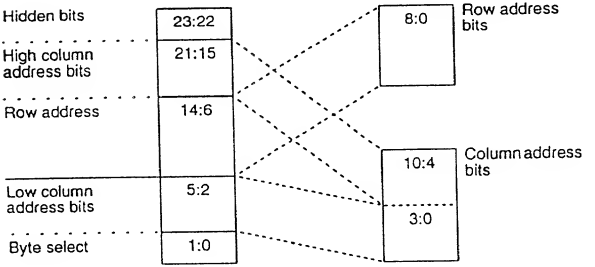

Hidden bits
23:22
High column address bits
21:15
Row address
14:6
Low column address bits
5:2
Byte select
1:0
8:0
Row address bits
10:4
Column address bits
3:0

FIG.48

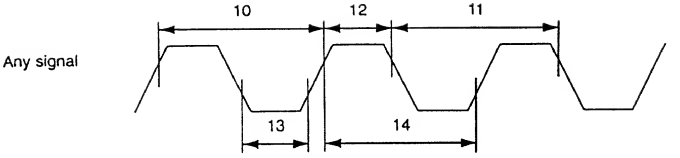

Any signal
10
12
11
13
14

FIG.49

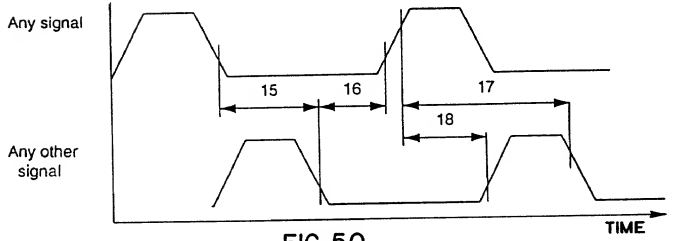

Any signal
15
16
17
18
Any other signal
TIME

FIG.50

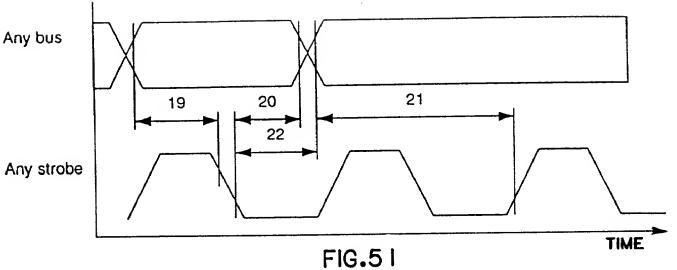

Any bus
19
20
21
22
Any strobe
TIME

FIG.5 I

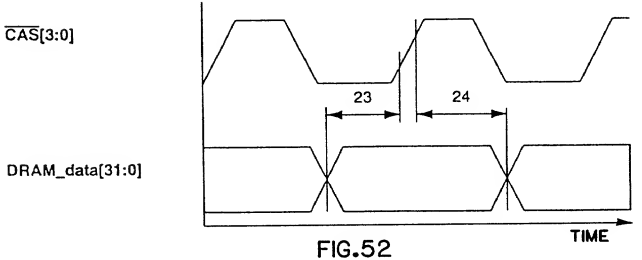

CAS[3:0]
23
24
DRAM_data[31:0]
TIME

FIG.52

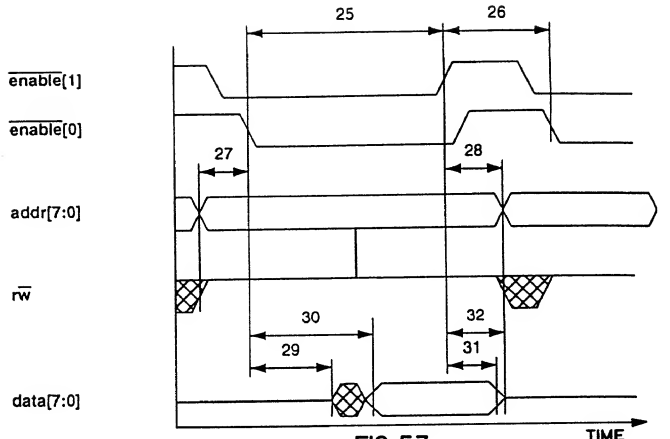

25
26
enable[1]
enable[0]
27
28
addr[7:0]
rw
30
32
29
31
data[7:0]
TIME

FIG.53

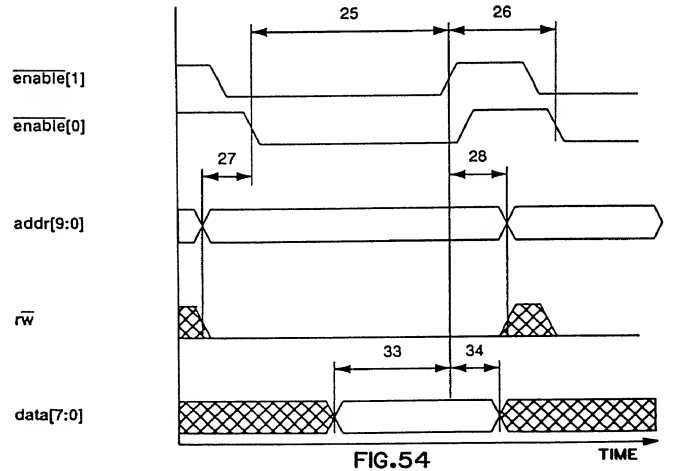

25
26
enable[1]
enable[0]
27
28
addr[9:0]
rw
33
34
data[7:0]
TIME

FIG.54

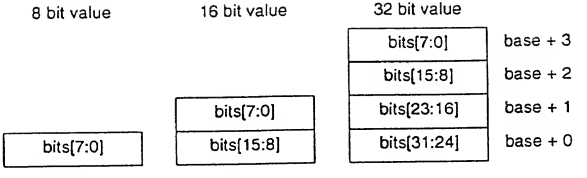

8 bit value
16 bit value
32 bit value
bits[7:0]
bits[7:0]
bits[15:8]
bits[7:0]
bits[15:8]
bits[23:16]
bits[31:24]
base + 3
base + 2
base + 1
base + 0

FIG.55

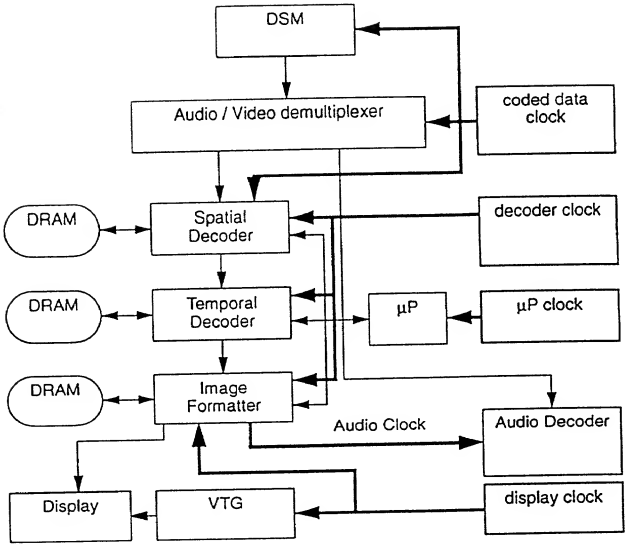

DSM
Audio / Video demultiplexer
coded data clock
DRAM
Spatial Decoder
decoder clock
DRAM
Temporal Decoder
µP
µP clock
DRAM
Image Formatter
Audio Clock
Audio Decoder
Display
VTG
display clock

FIG.56

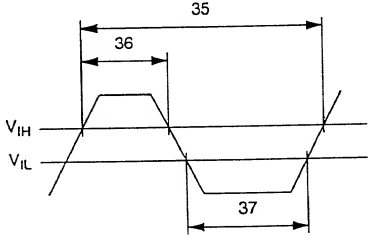

35
36
V_{IH}
V_{IL}
37

FIG.57

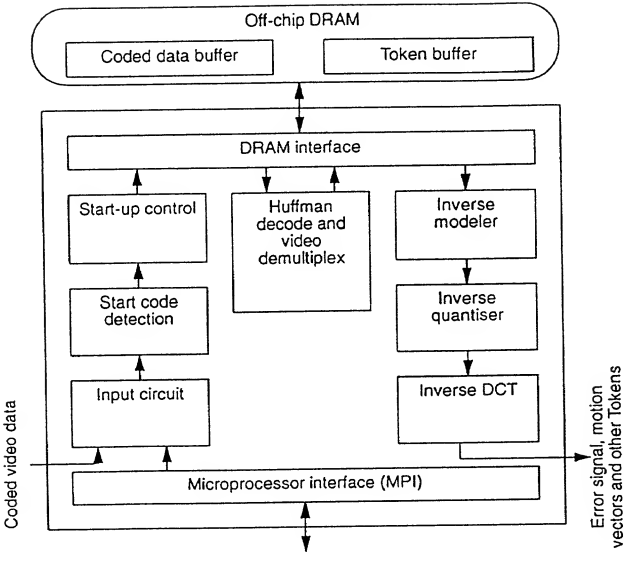

Off-chip DRAM
Coded data buffer
Token buffer
DRAM interface
Start-up control
Huffman decode and video demultiplex
Inverse modeler
Start code detection
Inverse quantiser
Input circuit
Inverse DCT
Coded video data
Microprocessor interface (MPI)
Error signal, motion vectors and other Tokens

FIG.58

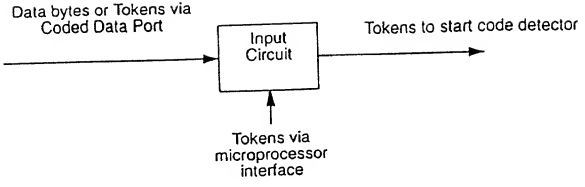

Data bytes or Tokens via
Coded Data Port
Input
Circuit
Tokens to start code detector
Tokens via
microprocessor
interface

FIG.59

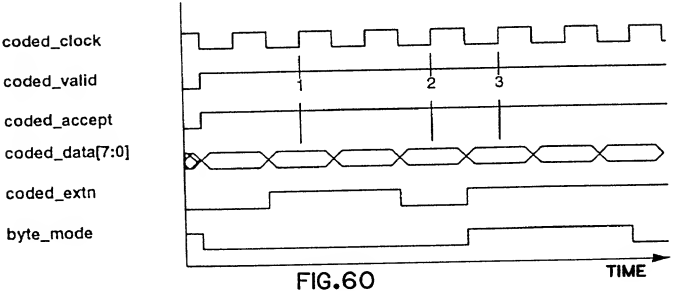

coded_clock
coded_valid
coded_accept
coded_data[7:0]
coded_extn
byte_mode
1
2
3
TIME

FIG.60

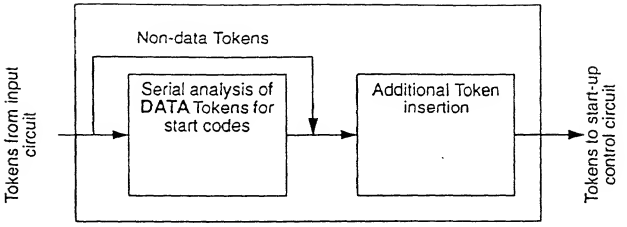

Non-data Tokens
Tokens from input circuit
Serial analysis of DATA Tokens for start codes
Additional Token insertion
Tokens to start-up control circuit

FIG.6 I

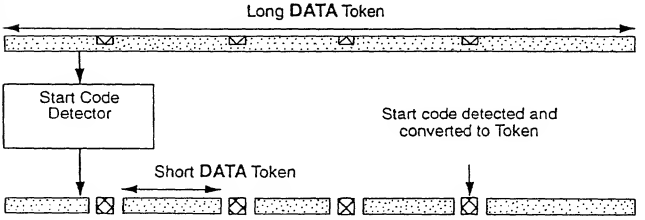

Long DATA Token
Start Code Detector
Start code detected and converted to Token
Short DATA Token

FIG.62

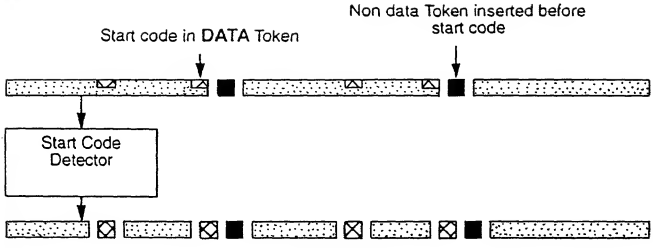

Non data Token inserted before start code
Start code in DATA Token
Start Code Detector

FIG.63

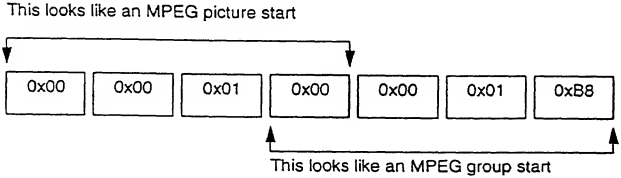

This looks like an MPEG picture start
0x00
0x00
0x01
0x00
0x00
0x01
0xB8
This looks like an MPEG group start

FIG.64

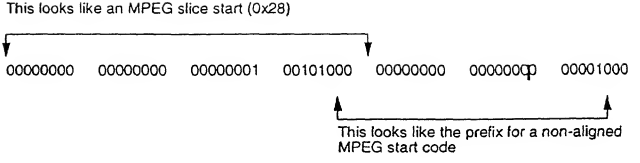

This looks like an MPEG slice start (0x28)
00000000 00000000 00000001 00101000 00000000 00000000 00001000
This looks like the prefix for a non-aligned MPEG start code

FIG.65

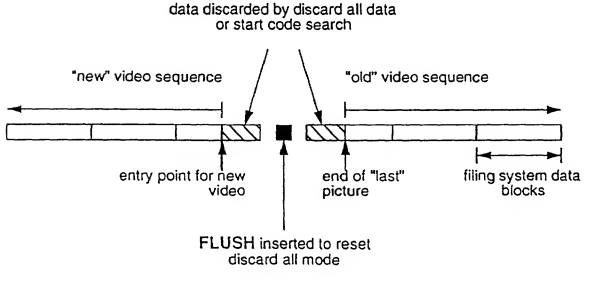

data discarded by discard all data or start code search
"new" video sequence
"old" video sequence
entry point for new video
end of "last" picture
filing system data blocks
FLUSH inserted to reset discard all mode

FIG.66

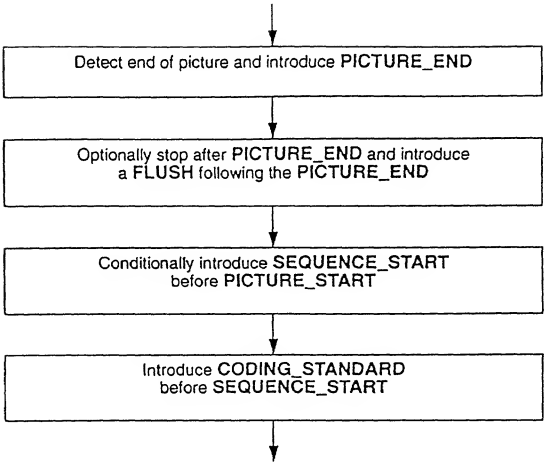

Detect end of picture and introduce PICTURE_END
Optionally stop after PICTURE_END and introduce a FLUSH following the PICTURE_END
Conditionally introduce SEQUENCE_START before PICTURE_START
Introduce CODING_STANDARD before SEQUENCE_START

FIG.67

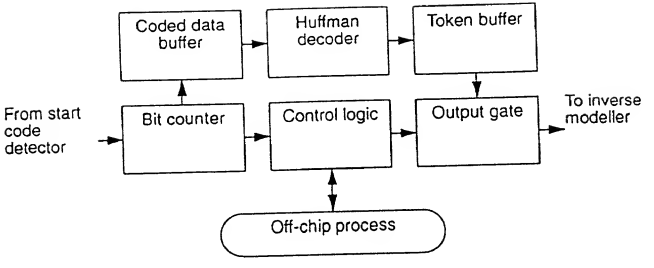

Coded data buffer
Huffman decoder
Token buffer
From start code detector
Bit counter
Control logic
Output gate
To inverse modeller
Off-chip process

FIG.68

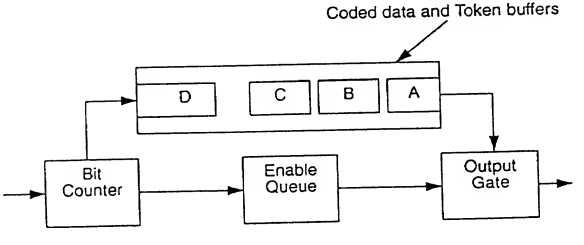

Coded data and Token buffers
D
C
B
A
Bit Counter
Enable Queue
Output Gate

FIG.69

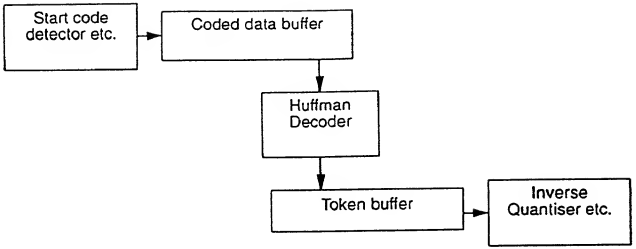

Start code detector etc.
Coded data buffer
Huffman Decoder
Token buffer
Inverse Quantiser etc.

FIG.70

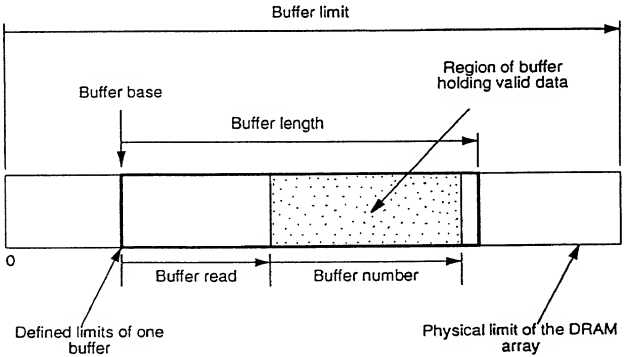

Buffer limit
Region of buffer holding valid data
Buffer base
Buffer length
0
Buffer read
Buffer number
Defined limits of one buffer
Physical limit of the DRAM array

FIG.71

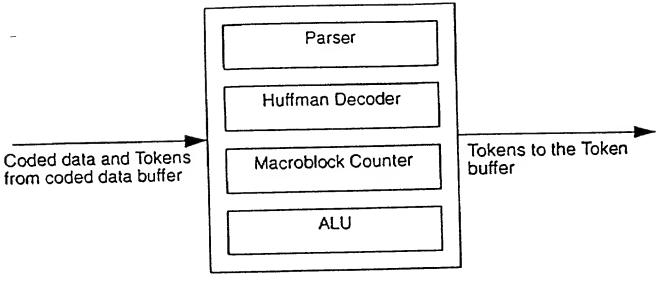

Parser
Huffman Decoder
Macroblock Counter
ALU
Coded data and Tokens from coded data buffer
Tokens to the Token buffer

FIG.72

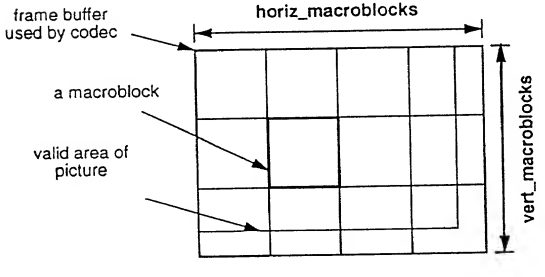

frame buffer used by codec
horiz_macroblocks
a macroblock
valid area of picture
vert_macroblocks

FIG.73

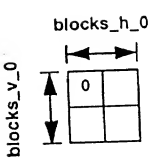


FIG.74A

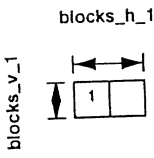


FIG.74B

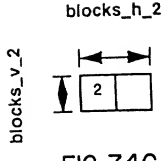


FIG.74C

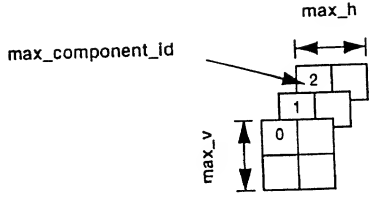


FIG.74D

$$\begin{cases} \text{horiz\_macroblocks} = \dfrac{\text{horiz\_pels} + 15}{16} \\ \text{vert\_macroblocks} = \dfrac{\text{vert\_pels} + 15}{16} \end{cases}$$

FIG.75

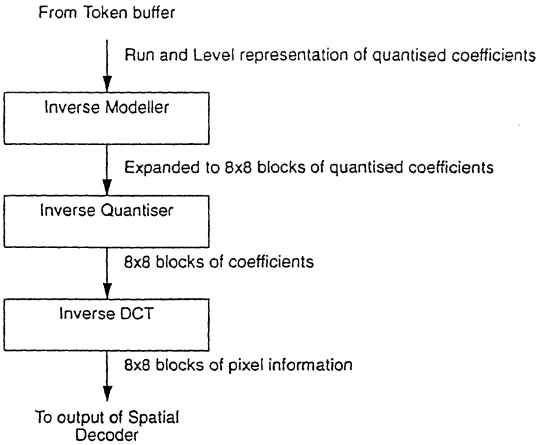

From Token buffer
Run and Level representation of quantised coefficients
Inverse Modeller
Expanded to 8x8 blocks of quantised coefficients
Inverse Quantiser
8x8 blocks of coefficients
Inverse DCT
8x8 blocks of pixel information
To output of Spatial Decoder

FIG.76

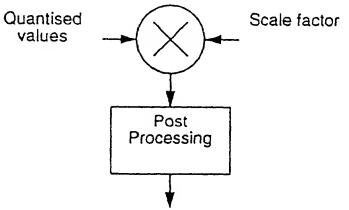

Quantised values
Scale factor
Post Processing

FIG.77

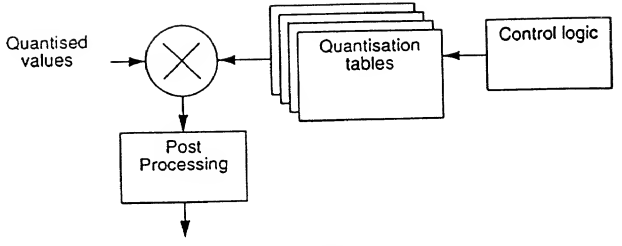

Quantised values
Quantisation tables
Control logic
Post Processing

FIG.78

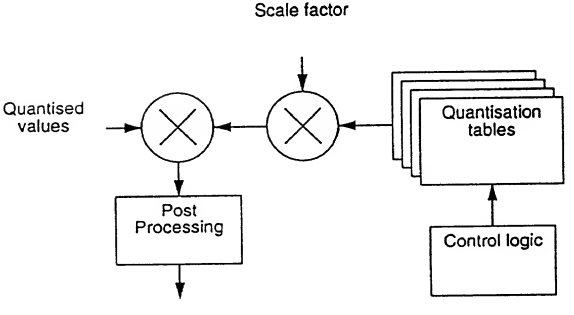

Scale factor
Quantised values
Quantisation tables
Post Processing
Control logic

FIG.79

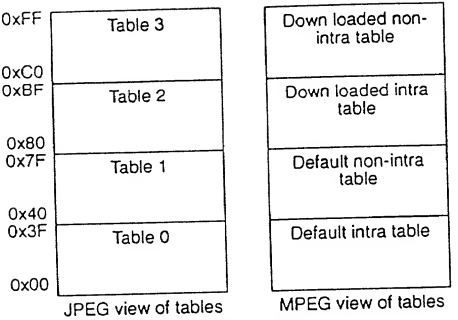


FIG.80

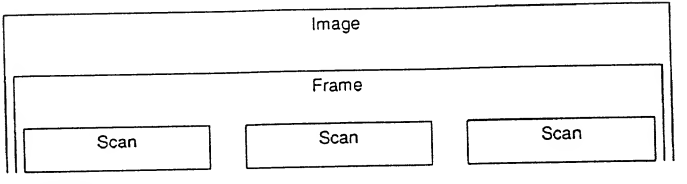


FIG.81

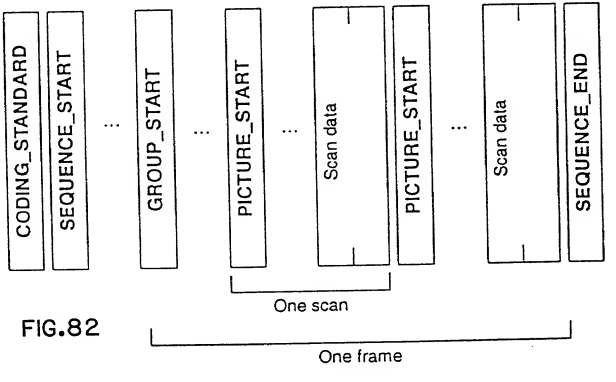


FIG.82

Off-chip DRAM
DRAM interface
Form prediction address
Interpolate / filter prediction
Separate motion vectors
FIFO
Picture sequence control
Microprocessor interface (μPI)
Error signal, motion vectors and other Tokens
Block organised video data and other Tokens

FIG.83

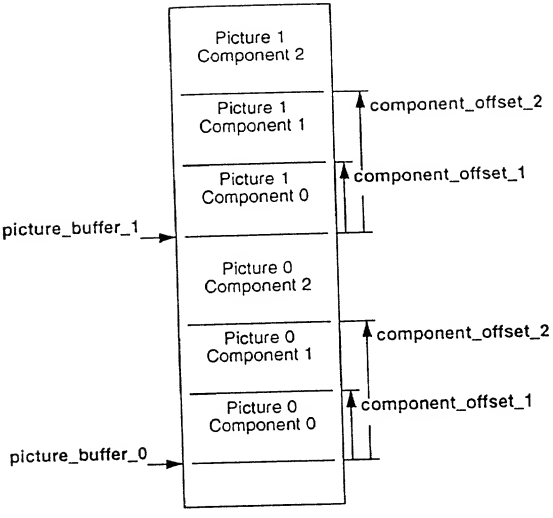

Picture 1
Component 2
Picture 1
Component 1
Picture 1
Component 0
picture_buffer_1
Picture 0
Component 2
Picture 0
Component 1
Picture 0
Component 0
picture_buffer_0
component_offset_2
component_offset_1
component_offset_2
component_offset_1

FIG.84

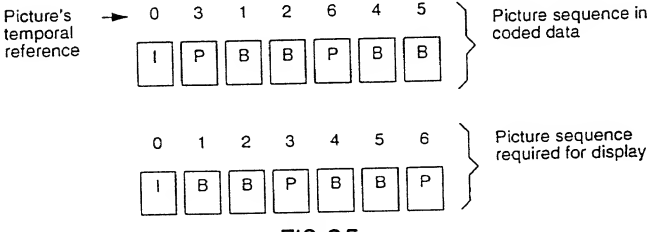

Picture's
temporal
reference
0 3 1 2 6 4 5
I P B B P B B
Picture sequence in
coded data
0 1 2 3 4 5 6
I B B P B B P
Picture sequence
required for display

FIG.85

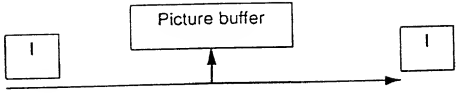


FIG.86

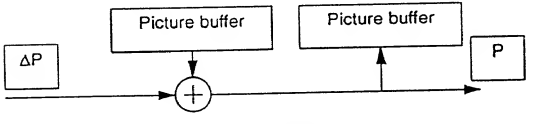


FIG.87

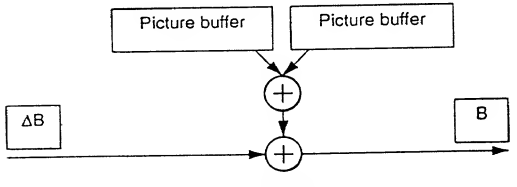


FIG.88

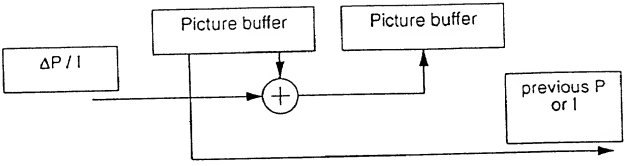

Picture buffer
Picture buffer
ΔP / I
previous P
or I

FIG.89

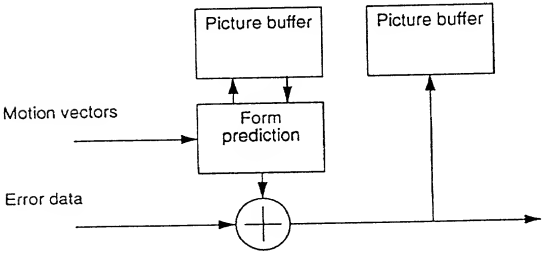

Picture buffer
Picture buffer
Motion vectors
Form
prediction
Error data

FIG.90

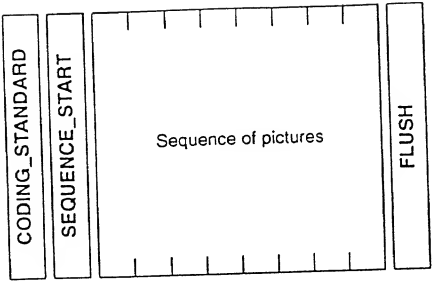

CODING_STANDARD
SEQUENCE_START
Sequence of pictures
FLUSH

FIG.91

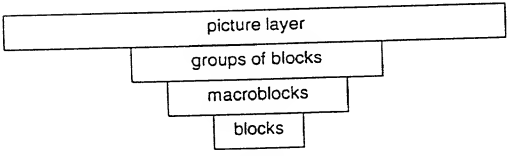

picture layer
groups of blocks
macroblocks
blocks

FIG.92

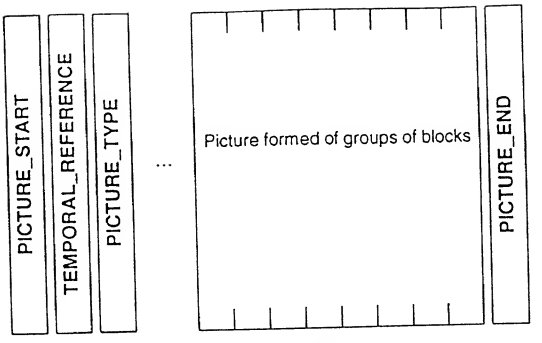

PICTURE_START
TEMPORAL_REFERENCE
PICTURE_TYPE
...
Picture formed of groups of blocks
PICTURE_END

FIG.93

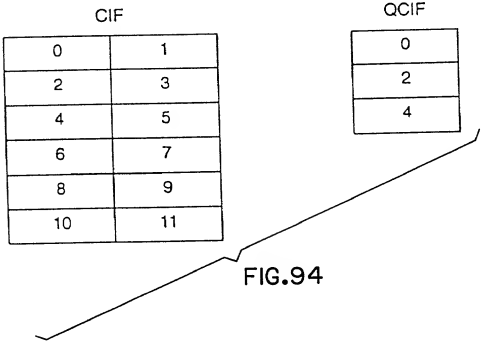

CIF
0
1
2
3
4
5
6
7
8
9
10
11
QCIF
0
2
4

FIG.94

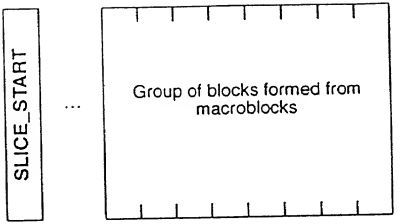


FIG.95

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33

FIG.96

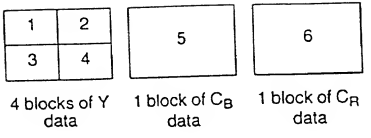

1
2
3
4
5
6
4 blocks of Y data
1 block of C_B data
1 block of C_R data

FIG.97

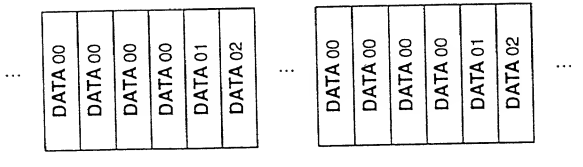

DATA 00
DATA 00
DATA 00
DATA 00
DATA 01
DATA 02
DATA 00
DATA 00
DATA 00
DATA 00
DATA 01
DATA 02

FIG.98

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

.
.
.

59	58	59	60	61	62	63	64

FIG.99

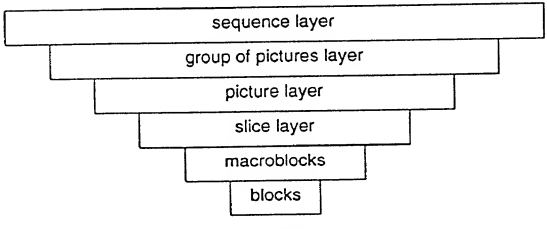


FIG. 100

FIG. 101

FIG. 102

FIG. 103

FIG. 104

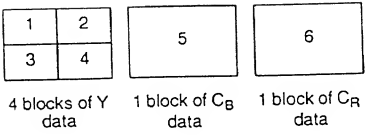


FIG. 105

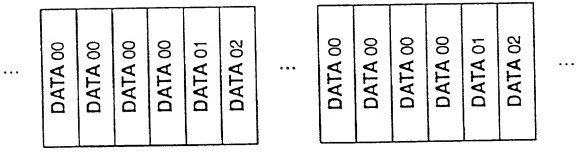


FIG. 106

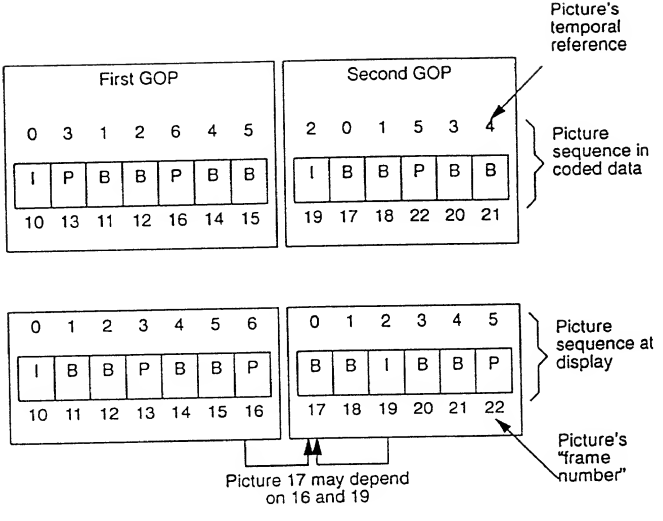

Picture's temporal reference
First GOP
Second GOP
0 3 1 2 6 4 5
I P B B P B B
10 13 11 12 16 14 15
2 0 1 5 3 4
I B B P B B
19 17 18 22 20 21
Picture sequence in coded data
0 1 2 3 4 5 6
I B B P B B P
10 11 12 13 14 15 16
0 1 2 3 4 5
B B I B B P
17 18 19 20 21 22
Picture sequence at display
Picture's "frame number"
Picture 17 may depend on 16 and 19

FIG. 107

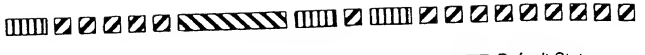


FIG. 108

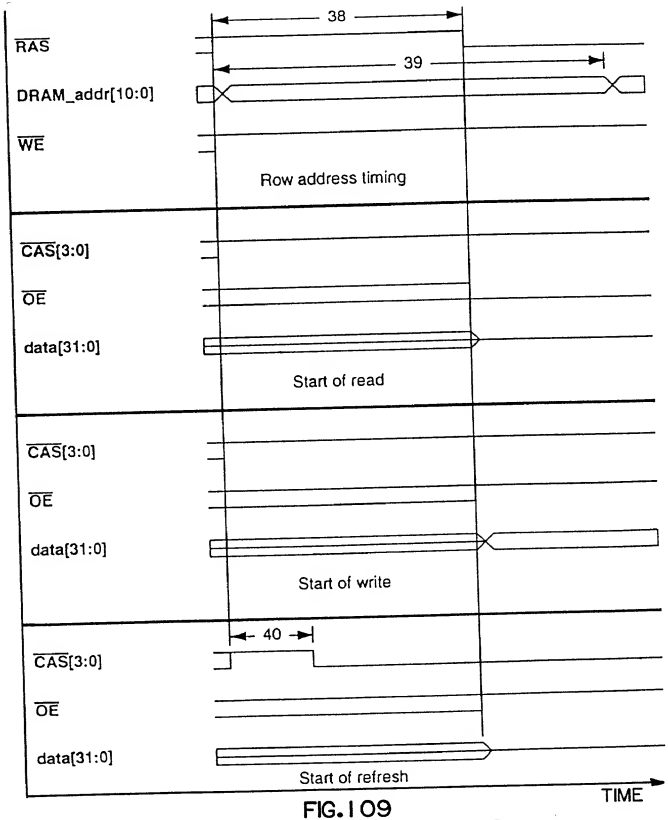


FIG. 109

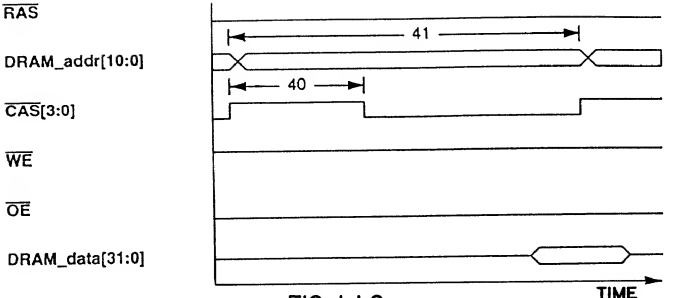


FIG. 110

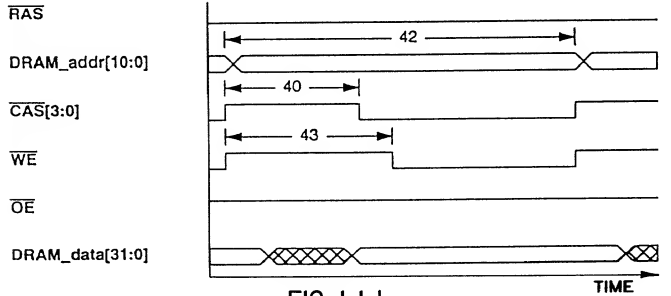


FIG. 111

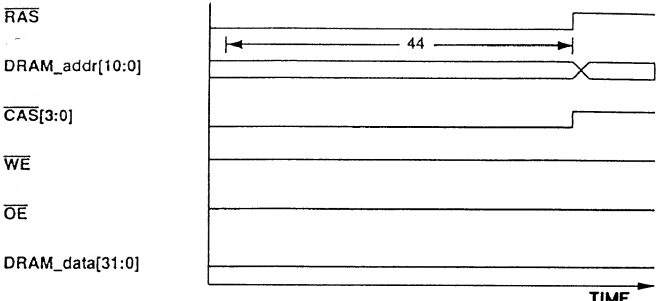


FIG. 1 12

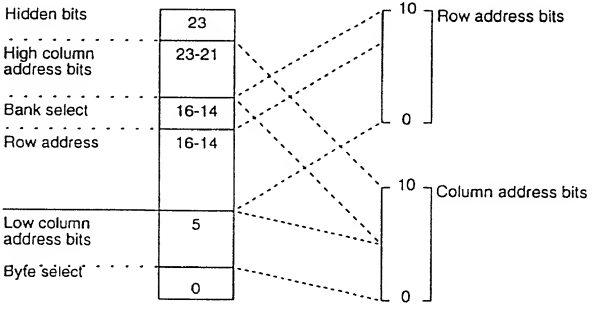


FIG. 1 13

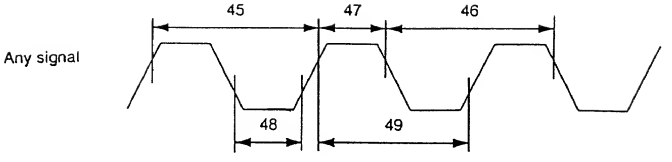


FIG. 114

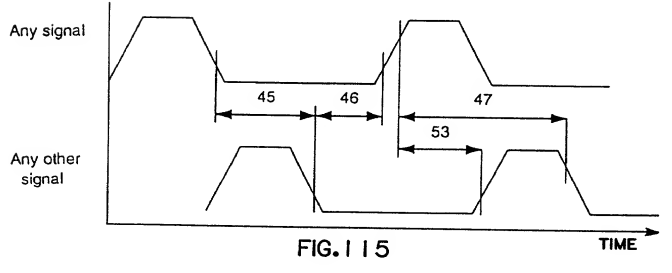


FIG. 115

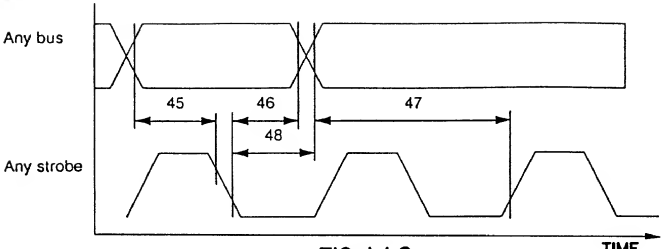


FIG. 116

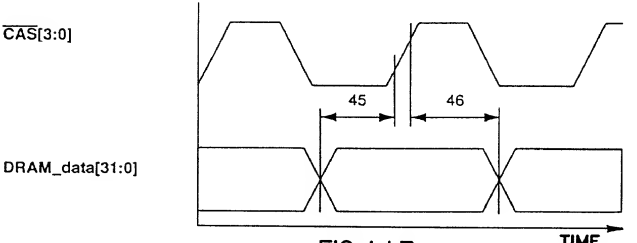


FIG. 117

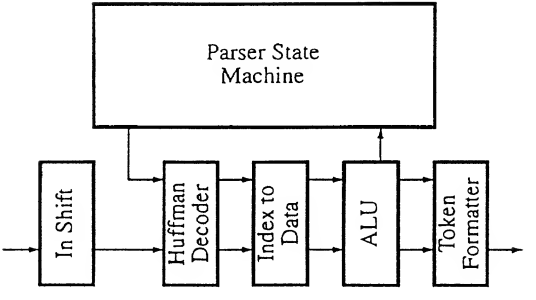

Parser State Machine
In Shift
Huffman Decoder
Index to Data
ALU
Token Formatter

FIG. 1 1 8

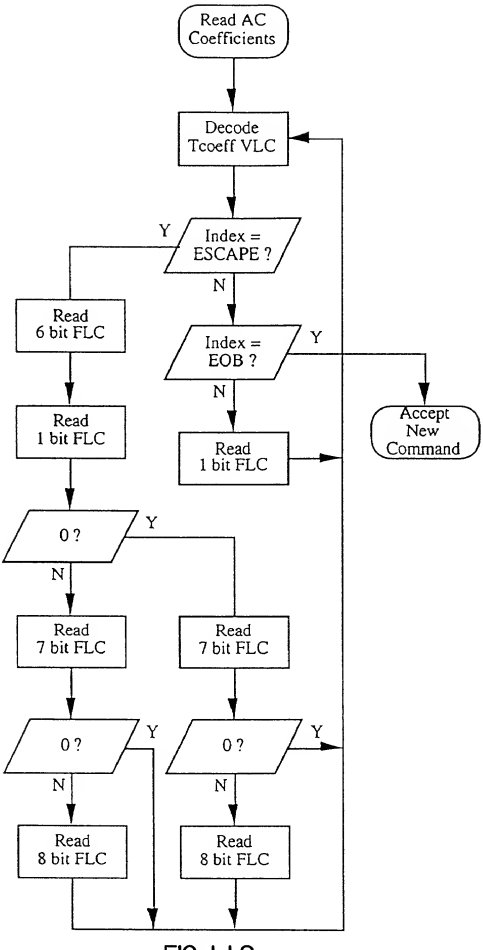

Read AC Coefficients
Decode Tcoeff VLC
Index = ESCAPE ?
Y
N
Read 6 bit FLC
Index = EOB ?
Y
N
Accept New Command
Read 1 bit FLC
Read 1 bit FLC
0 ?
Y
N
Read 7 bit FLC
Read 7 bit FLC
0 ?
Y
0 ?
Y
N
N
Read 8 bit FLC
Read 8 bit FLC

FIG. 1 1 9

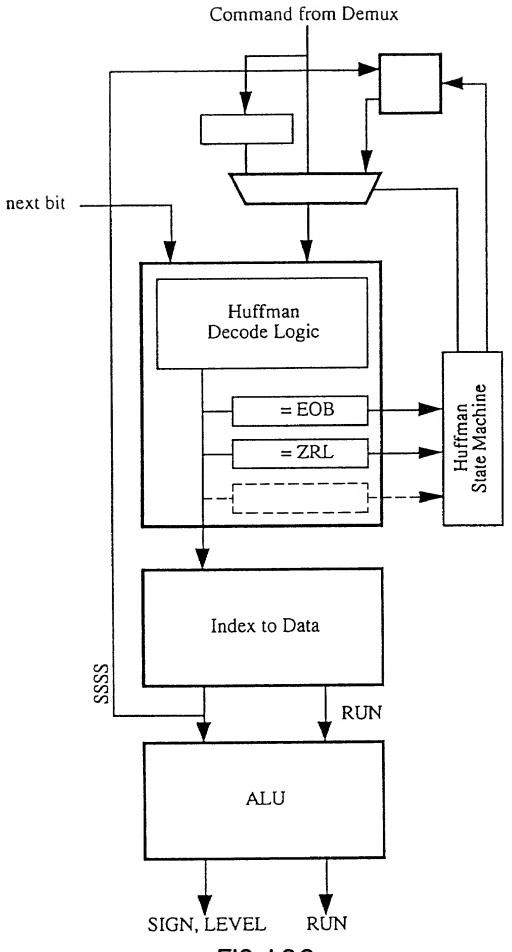

Command from Demux
next bit
Huffman
Decode Logic
= EOB
= ZRL
Huffman
State Machine
Index to Data
SSSS
RUN
ALU
SIGN, LEVEL
RUN

FIG. 120

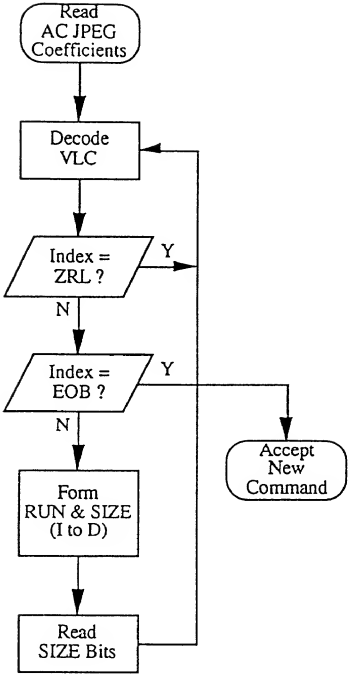


FIG. 121A

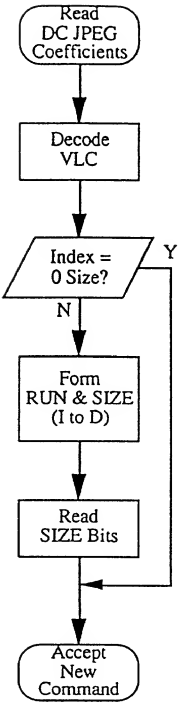


FIG. 121B

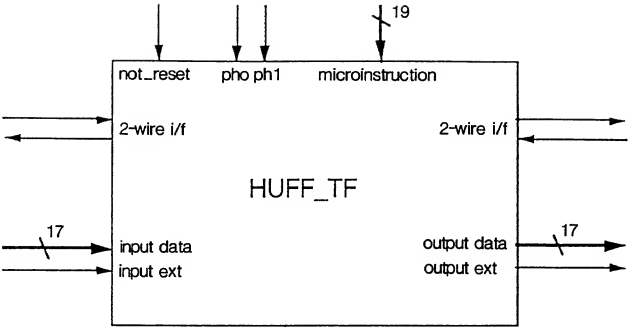


FIG. 122

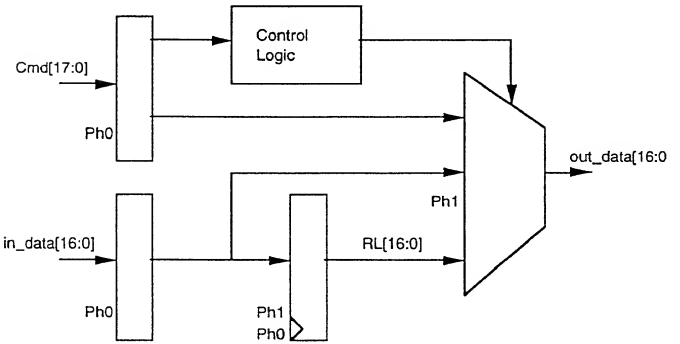


FIG. 123

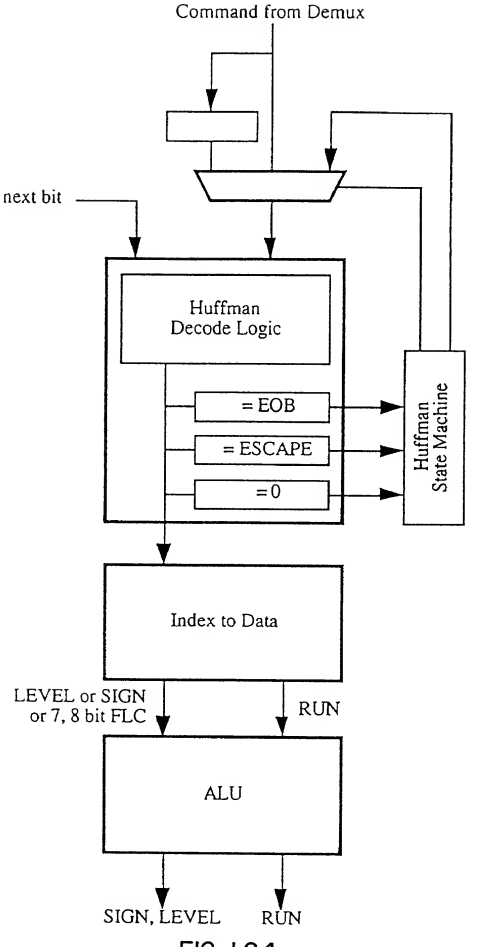

Command from Demux
next bit
Huffman
Decode Logic
= EOB
= ESCAPE
= 0
Huffman
State Machine
Index to Data
LEVEL or SIGN
or 7, 8 bit FLC
RUN
ALU
SIGN, LEVEL
RUN

FIG. 124

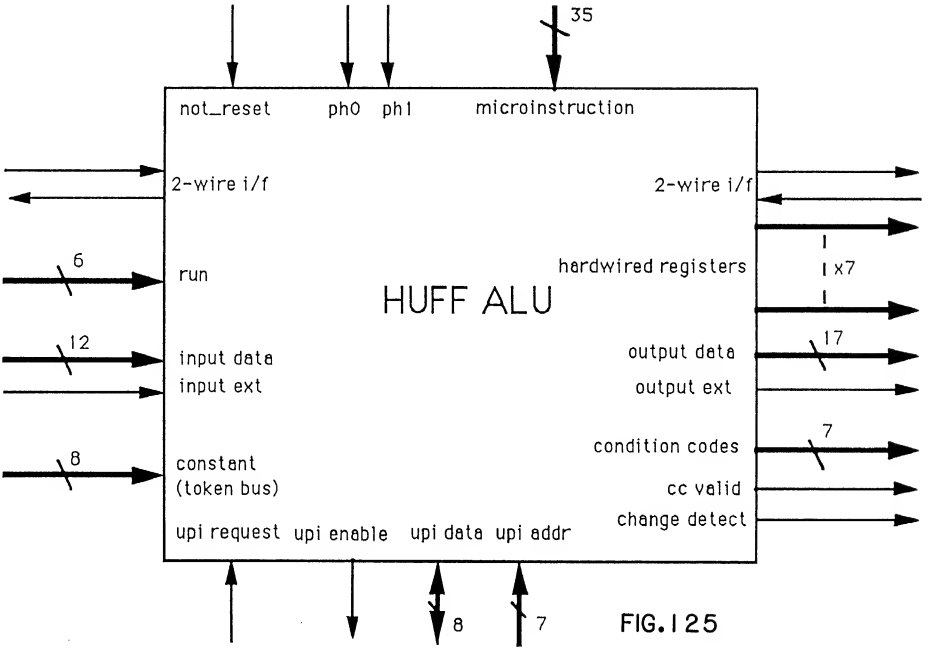

35
not_reset
ph0
ph1
microinstruction
2-wire i/f
2-wire i/f
6
run
hardwired registers
x7
HUFF ALU
12
input data
input ext
output data
17
output ext
8
constant
(token bus)
condition codes
7
cc valid
change detect
upi request
upi enable
upi data
upi addr
8
7

FIG. 125

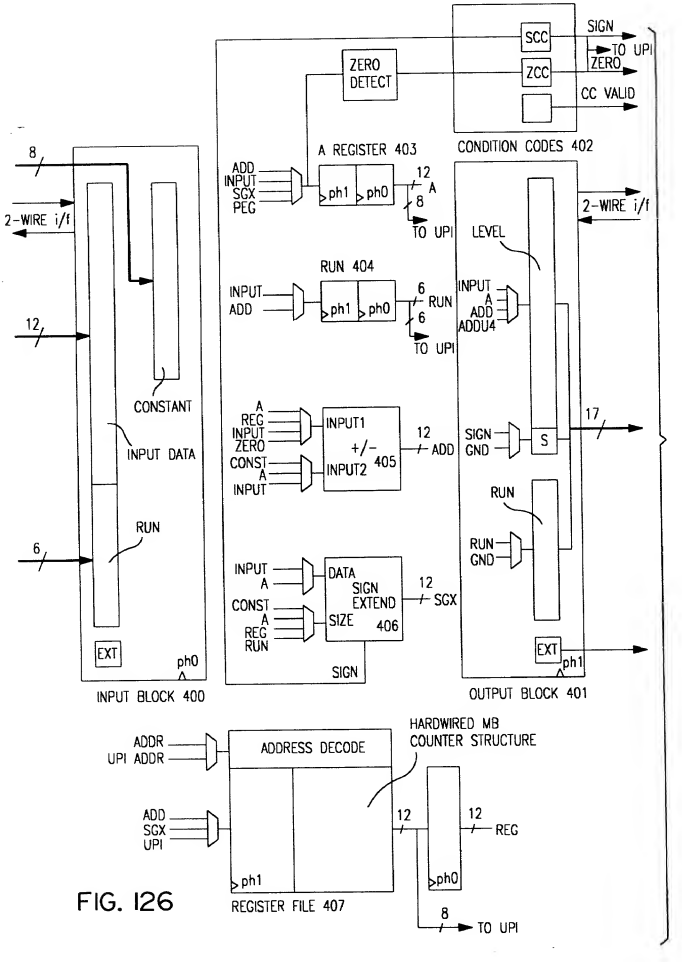

SCC
SIGN
TO UPI
ZERO DETECT
ZCC
ZERO
CC VALID
CONDITION CODES 402
A REGISTER 403
ADD
INPUT
SGX
PEG
ph1
ph0
12
A
8
TO UPI
8
2-WIRE i/f
2-WIRE i/f
LEVEL
RUN 404
INPUT
ADD
6
RUN
6
TO UPI
INPUT
A
ADD
ADDU4
12
CONSTANT
INPUT DATA
A
REG
INPUT
ZERO
CONST
A
INPUT
INPUT1
+/-
INPUT2
405
12
ADD
SIGN
GND
S
17
RUN
6
RUN
RUN
GND
INPUT
A
CONST
A
REG
RUN
DATA
SIGN
EXTEND
SIZE
406
12
SGX
SIGN
EXT
ph0
EXT
ph1
INPUT BLOCK 400
OUTPUT BLOCK 401
HARDWIRED MB
COUNTER STRUCTURE
ADDR
UPI ADDR
ADDRESS DECODE
ADD
SGX
UPI
12
12
REG
ph1
ph0
8
TO UPI
REGISTER FILE 407

FIG. 126

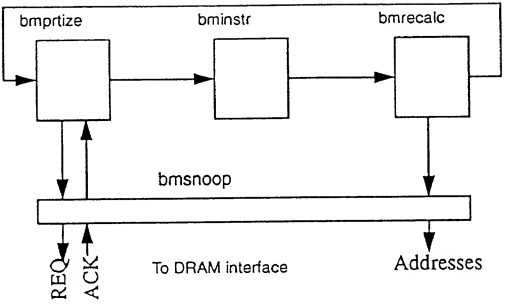

bmprtize
bminstr
bmrecalc
bmsnoop
REQ
ACK
To DRAM interface
Addresses

FIG. 127

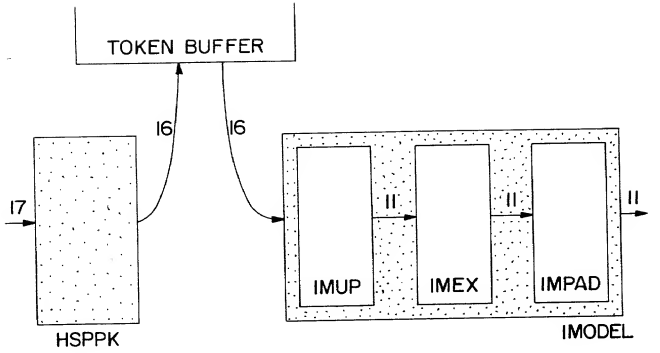


FIG. 128

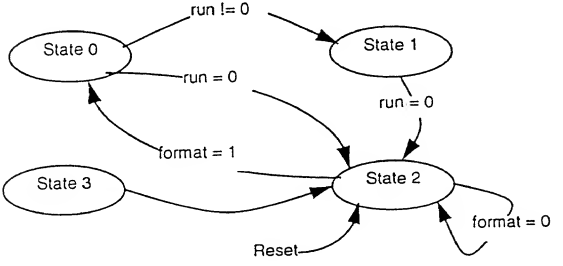


FIG. 129

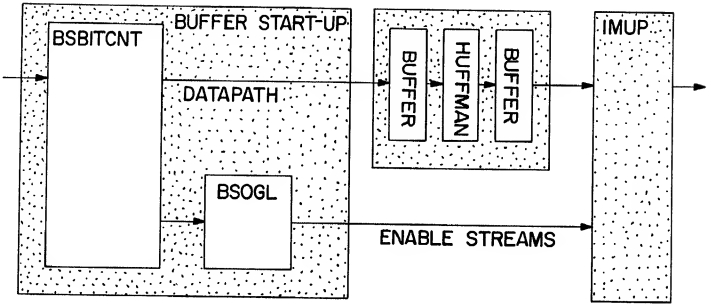


FIG. 130

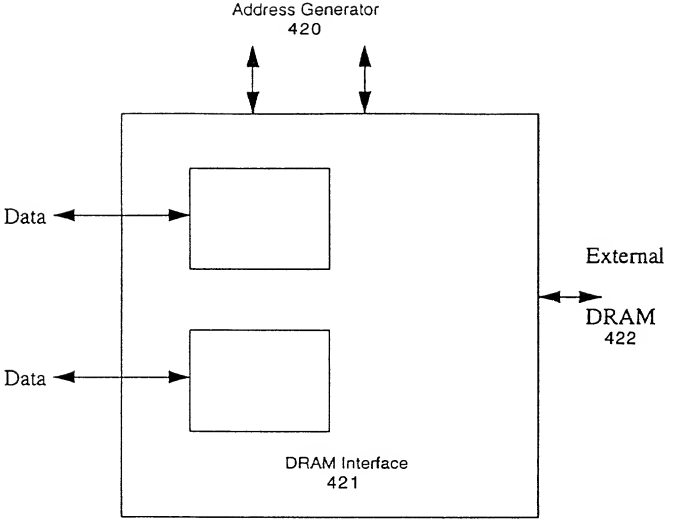

Address Generator
420
Data
Data
External
DRAM
422
DRAM Interface
421

FIG. 131

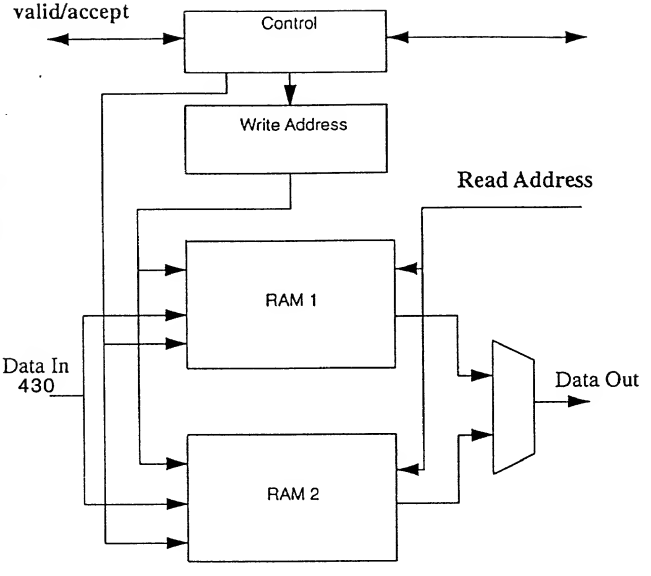

valid/accept
Control
Write Address
Read Address
RAM 1
Data In
430
Data Out
RAM 2

FIG. 132

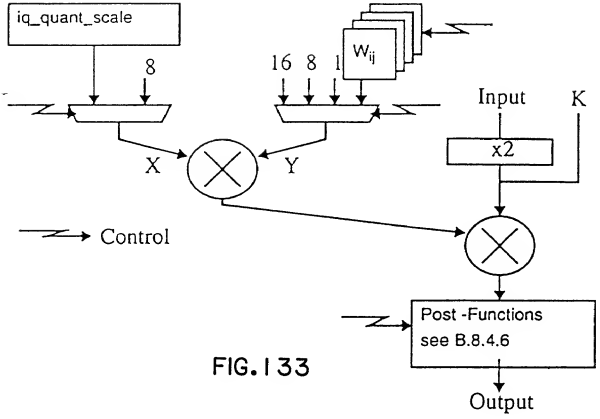


FIG. 133

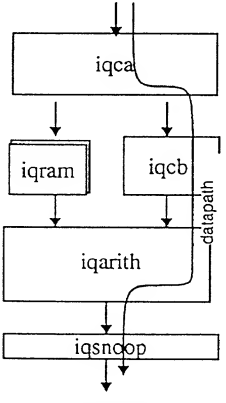


FIG. 134

BodyWord counter
63WordsInBody
TokenHeader
HeaderExtn
ExtnBit
PLA
NextSubstate
Address
Read/Write
Enable
InsertNext
Coefficients?

FIG. 135

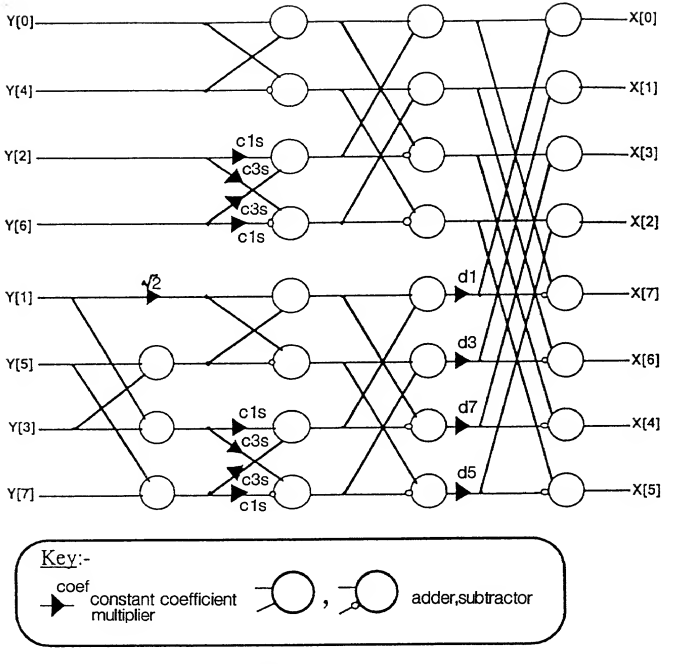

Y[0]
Y[4]
Y[2]
Y[6]
Y[1]
Y[5]
Y[3]
Y[7]
c1s
c3s
c3s
c1s
√2
c1s
c3s
c3s
c1s
d1
d3
d7
d5
X[0]
X[1]
X[3]
X[2]
X[7]
X[6]
X[4]
X[5]
Key:-
coef
constant coefficient multiplier
adder,subtractor

FIG. 136

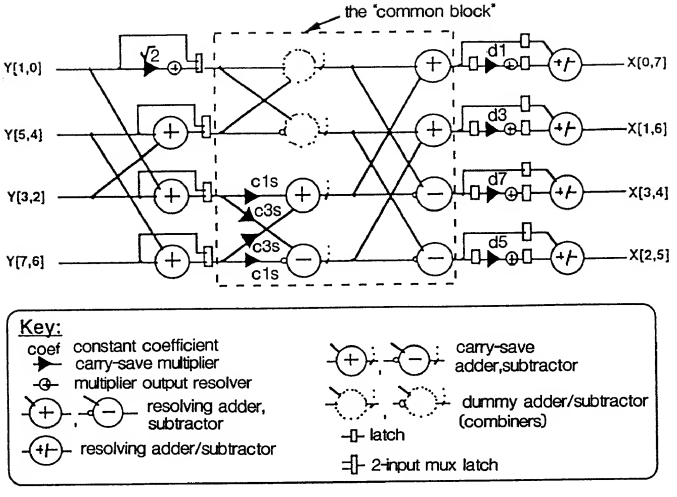


FIG. 137

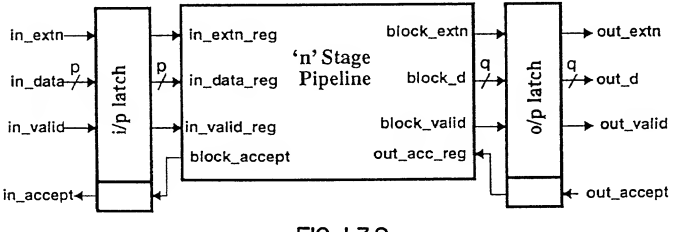


FIG. 138

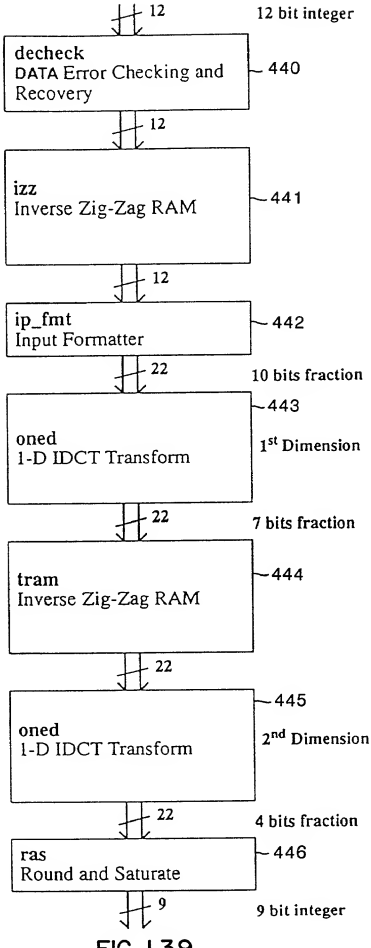

12
12 bit integer
decheck
DATA Error Checking and Recovery
440
12
izz
Inverse Zig-Zag RAM
441
12
ip_fmt
Input Formatter
442
22
10 bits fraction
443
oned
1-D IDCT Transform
1st Dimension
22
7 bits fraction
tram
Inverse Zig-Zag RAM
444
22
445
oned
1-D IDCT Transform
2nd Dimension
22
4 bits fraction
ras
Round and Saturate
446
9
9 bit integer

FIG. 139

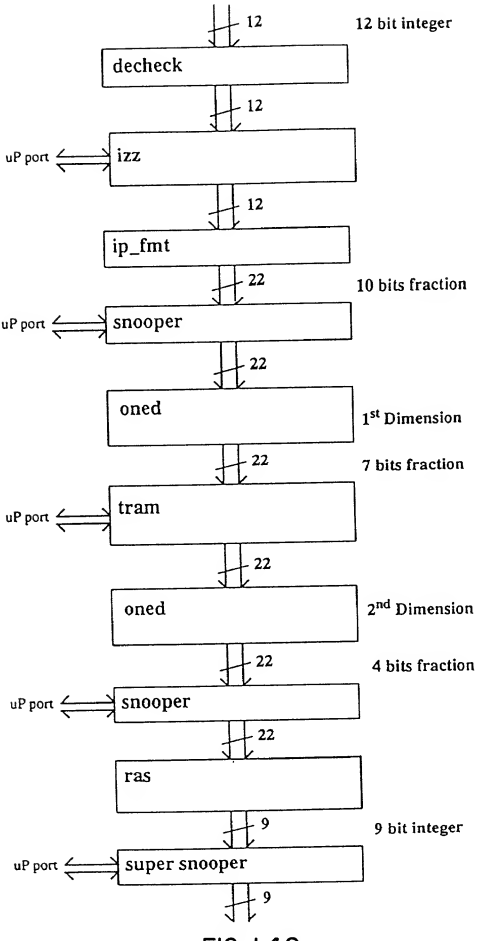
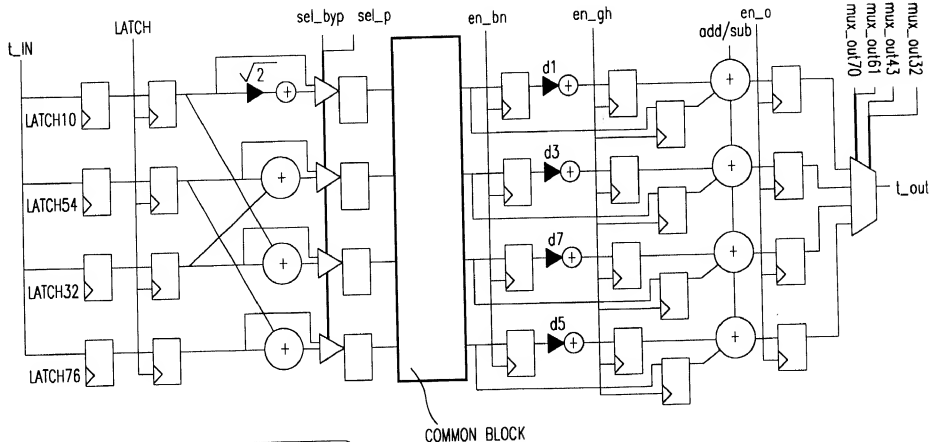

12
12 bit integer
decheck
12
uP port
izz
12
ip_fmt
22
10 bits fraction
uP port
snooper
22
oned
1st Dimension
22
7 bits fraction
uP port
tram
22
oned
2nd Dimension
22
4 bits fraction
uP port
snooper
22
ras
9
9 bit integer
uP port
super snooper
9

FIG. 140



NOTE: "COMMON BLOCK" IS ENTIRELY COMBINATIONAL (NO LATCHING)

FIG. 141

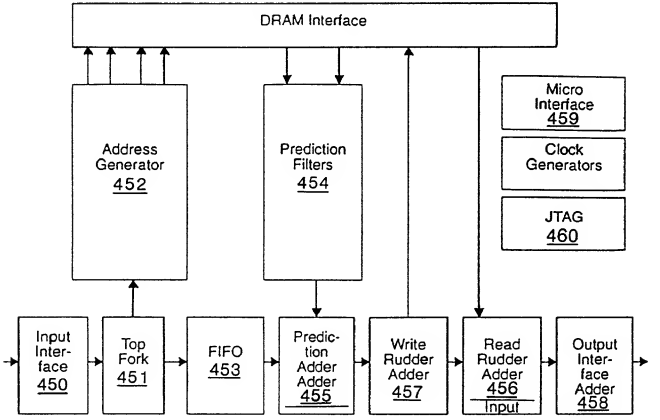

DRAM Interface
Address Generator 452
Prediction Filters 454
Micro Interface 459
Clock Generators
JTAG 460
Input Interface 450
Top Fork 451
FIFO 453
Predic-tion Adder Adder 455
Write Rudder Adder 457
Read Rudder Adder 456
Input
Output Inter-face Adder 458

FIG. 142

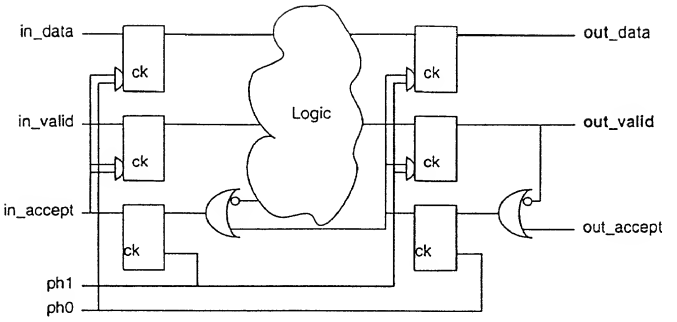

in_data
in_valid
in_accept
ph1
ph0
ck
Logic
out_data
out_valid
out_accept

FIG. 143

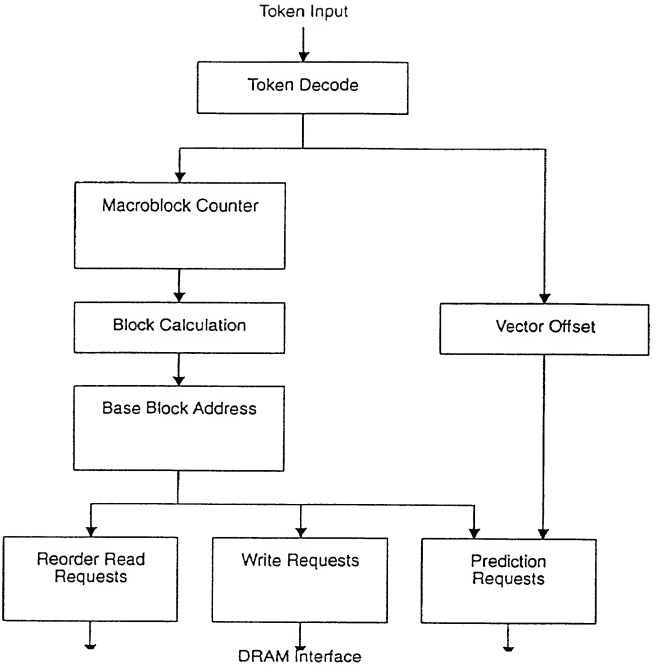

Token Input
Token Decode
Macroblock Counter
Block Calculation
Vector Offset
Base Block Address
Reorder Read Requests
Write Requests
Prediction Requests
DRAM Interface

FIG. 144

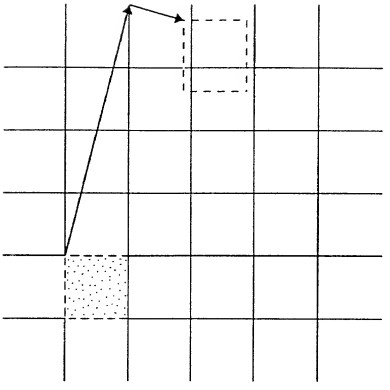


FIG. 145

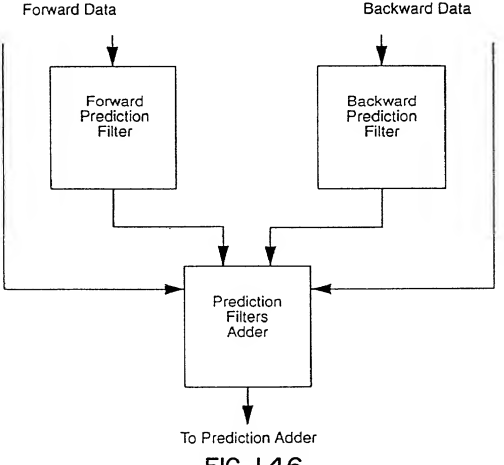


FIG. 146

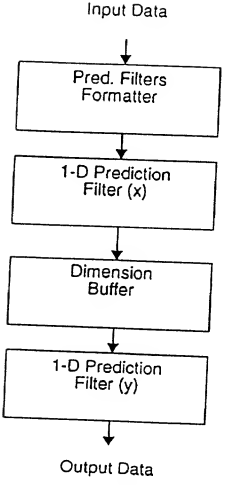

Input Data
Pred. Filters Formatter
1-D Prediction Filter (x)
Dimension Buffer
1-D Prediction Filter (y)
Output Data

FIG. 147

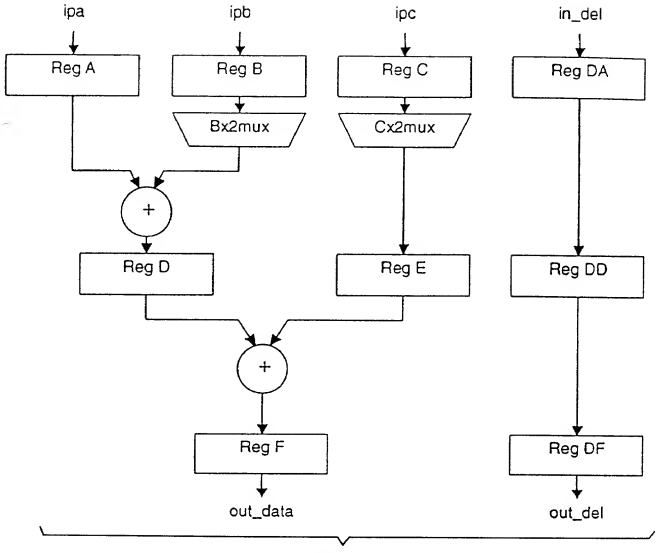


FIG. 148

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

FIG. 149

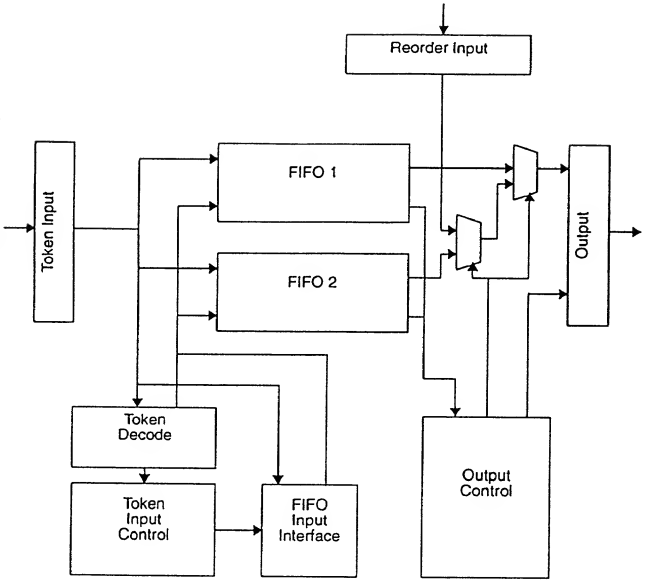

Reorder Input
Token Input
FIFO 1
FIFO 2
Output
Token Decode
Token Input Control
FIFO Input Interface
Output Control

FIG. 150

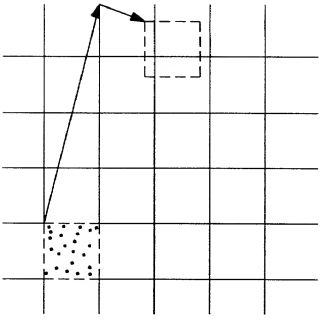


FIG. 151

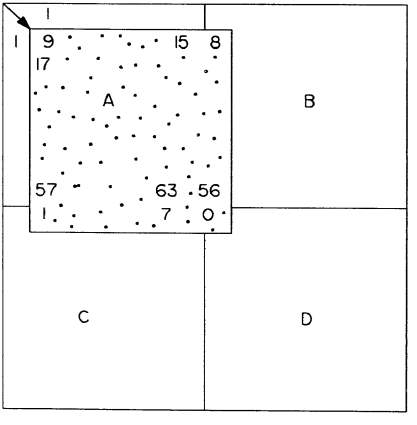


FIG. 152

Read Cycle

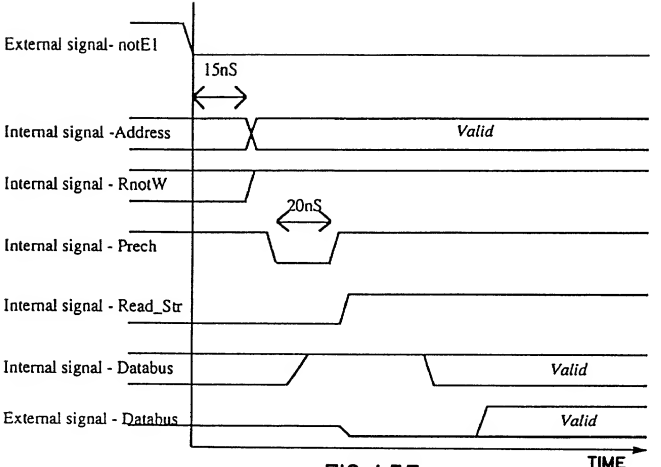


FIG. 153

Write Cycle

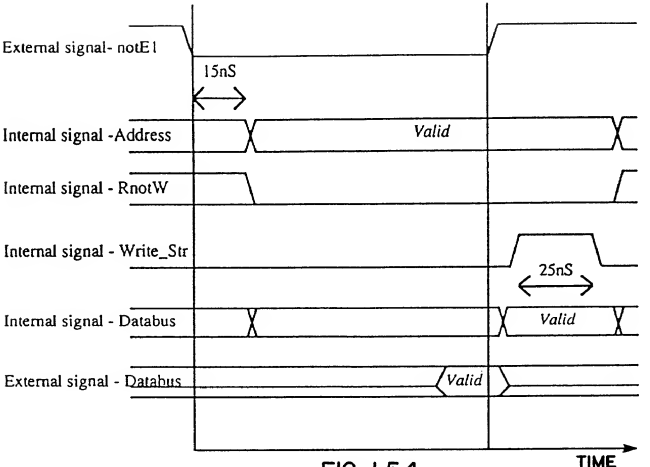


FIG. 154

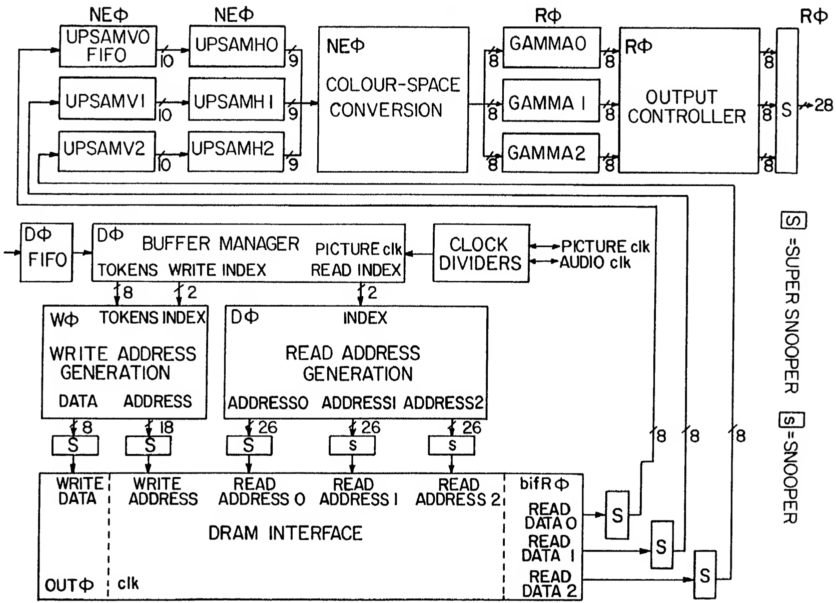

NEΦ
UPSAMVO FIFO
UPSAMVI
UPSAMV2
10
NEΦ
UPSAMHO
UPSAMH I
UPSAMH2
9
NEΦ
COLOUR-SPACE CONVERSION
RΦ
GAMMAO
GAMMA I
GAMMA2
8
RΦ
OUTPUT CONTROLLER
S
28
DΦ
FIFO
DΦ
BUFFER MANAGER
TOKENS
WRITE INDEX
PICTURE clk
READ INDEX
CLOCK DIVIDERS
PICTURE clk
AUDIO clk
WΦ
TOKENS
INDEX
WRITE ADDRESS GENERATION
DATA
ADDRESS
DΦ
INDEX
READ ADDRESS GENERATION
ADDRESSO
ADDRESSI
ADDRESS2
18
26
2
s
WRITE DATA
WRITE ADDRESS
READ ADDRESS O
READ ADDRESS I
READ ADDRESS 2
DRAM INTERFACE
OUTΦ
clk
bifRΦ
READ DATA O
READ DATA I
READ DATA 2
S =SUPER SNOOPER
s =SNOOPER

FIG. 155

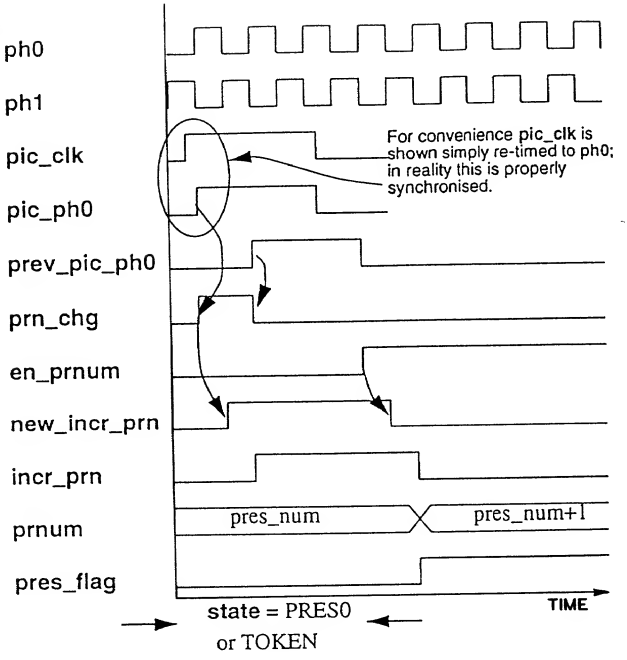

ph0
ph1
pic_clk
pic_ph0
prev_pic_ph0
prn_chg
en_prnum
new_incr_prn
incr_prn
prnum
pres_flag
For convenience pic_clk is shown simply re-timed to ph0; in reality this is properly synchronised.
pres_num
pres_num+1
state = PRES0
or TOKEN
TIME

FIG. 156

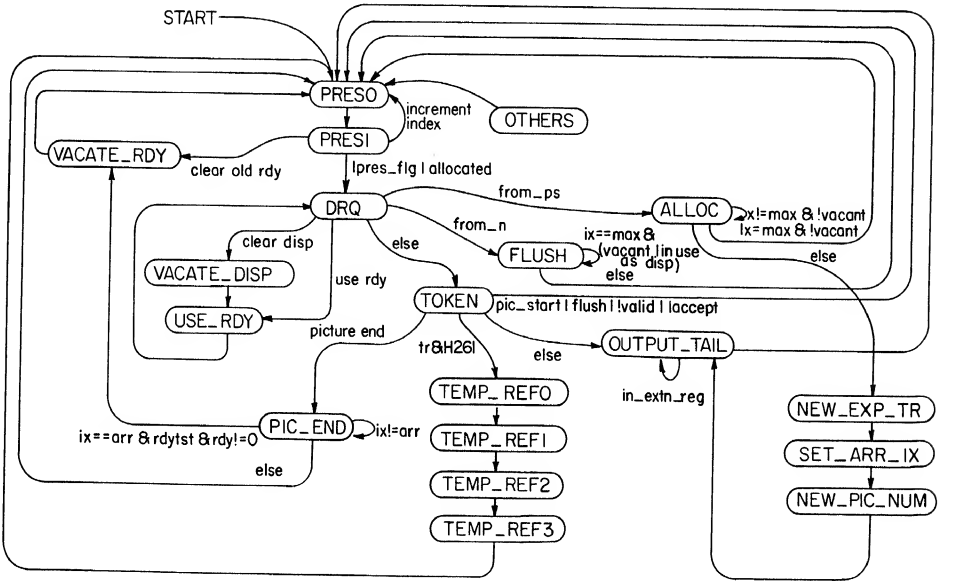

START
PRES0
increment index
OTHERS
PRES1
VACATE_RDY
clear old rdy
!pres_flg ! allocated
DRQ
from_ps
ALLOC
x!=max & !vacant
!x=max & !vacant
from_n
clear disp
else
ix==max & (vacant ! in use as disp)
FLUSH
else
VACATE_DISP
use rdy
TOKEN
pic_start ! flush ! !valid ! !accept
USE_RDY
picture end
tr8H26l
else
OUTPUT_TAIL
in_extn_reg
TEMP_REF0
TEMP_REF1
TEMP_REF2
TEMP_REF3
PIC_END
ix!=arr
ix==arr & rdytst & rdy!=0
else
NEW_EXP_TR
SET_ARR_IX
NEW_PIC_NUM

FIG. 157

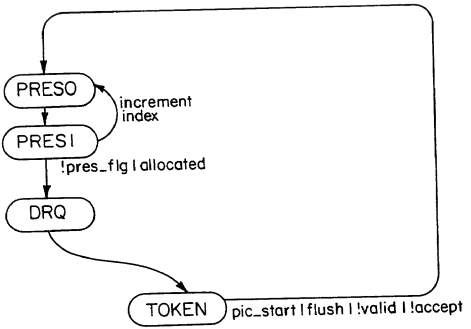

PRESO
increment
index
PRES I
!pres_flg I allocated
DRQ
TOKEN
pic_start I flush I !valid I !accept

FIG. I 58

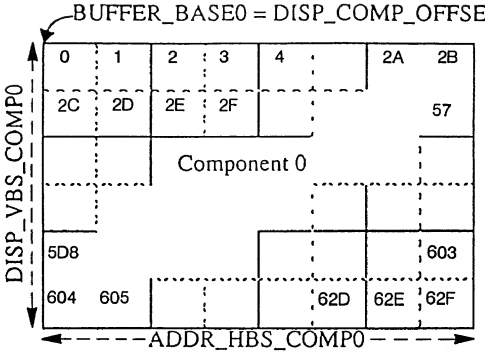


FIG. 159A

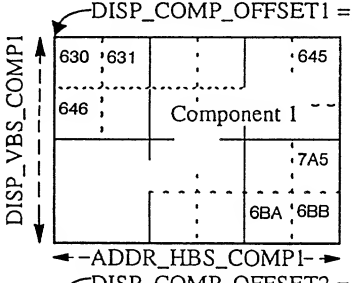


FIG. 159B

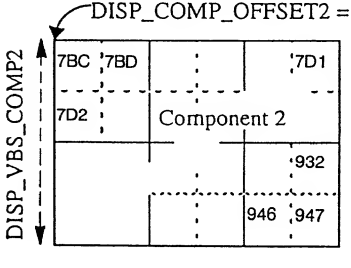


FIG. 159C

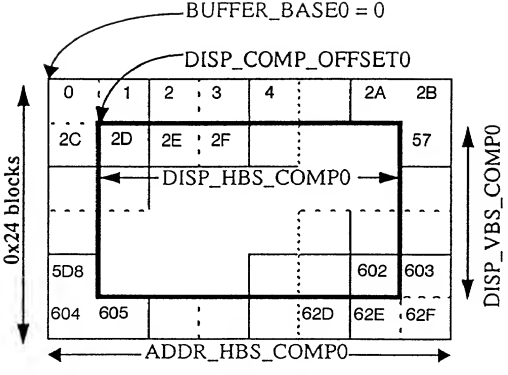

BUFFER_BASE0 = 0
DISP_COMP_OFFSET0
0
1
2
3
4
2A
2B
2C
2D
2E
2F
57
DISP_HBS_COMP0
0x24 blocks
DISP_VBS_COMP0
5D8
602
603
604
605
62D
62E
62F
ADDR_HBS_COMP0

FIG. 160

BUFFER OFFSET 0x00

COMPONENT OFFSET 0x000 + ...

00	01	02	03	04	05	06	07	08	09	0A	0B
0C	0D	0E	0F	10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B
3C	3D	3E	3F	40	41	42	43	44	45	46	47
48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B
6C	6D	6E	6F	70	71	72	73	74	75	76	77
78	79	7A	7B	7C	7D	7E	7F	80	81	82	83
84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

FIG. 16 1A

COMPONENT1 OFFSET 0x100 + ...

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

FIG. 16 1B

COMPONENT1 OFFSET 0x200 + ...

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

FIG. 16 1C

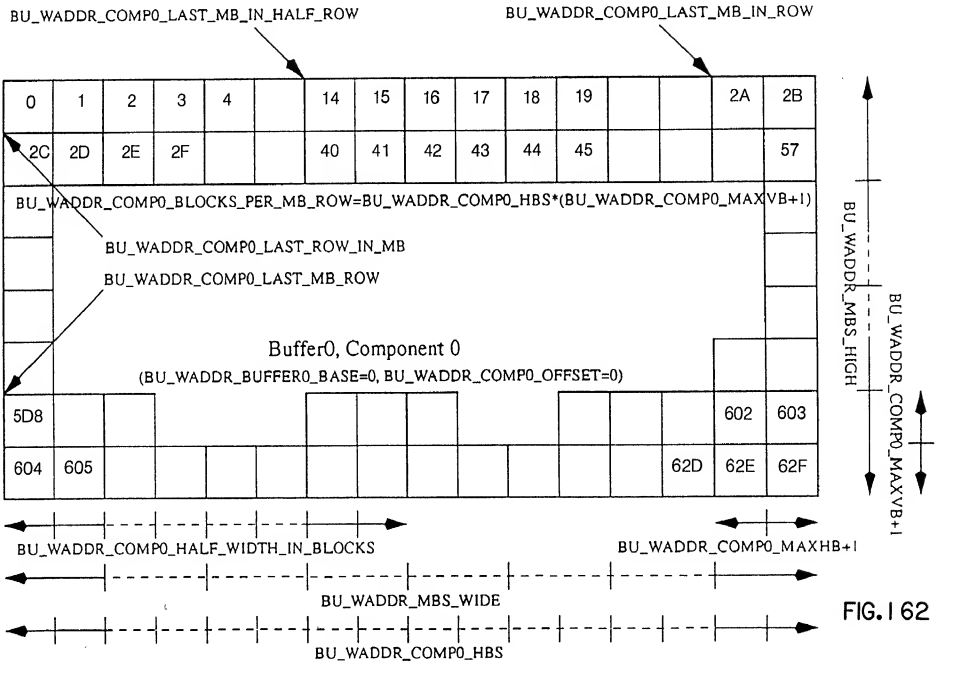
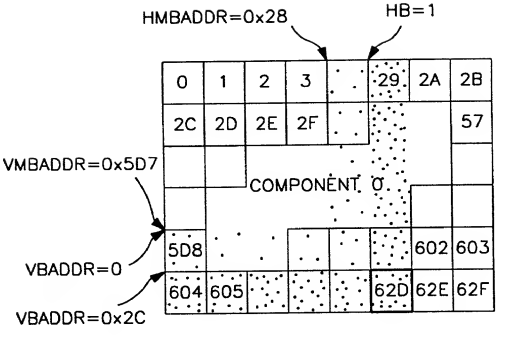
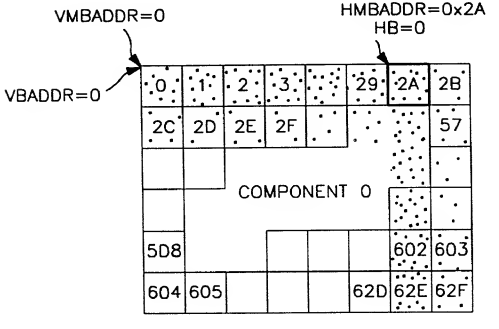


FIG. 162



BLOCK ADDRESS=0+0+0x5D8+0x28+0x2C+1=0x62D

FIG. 163A



BLOCK ADDRESS=0+0+0+0x2A+0+0=0x2A

FIG. 163B

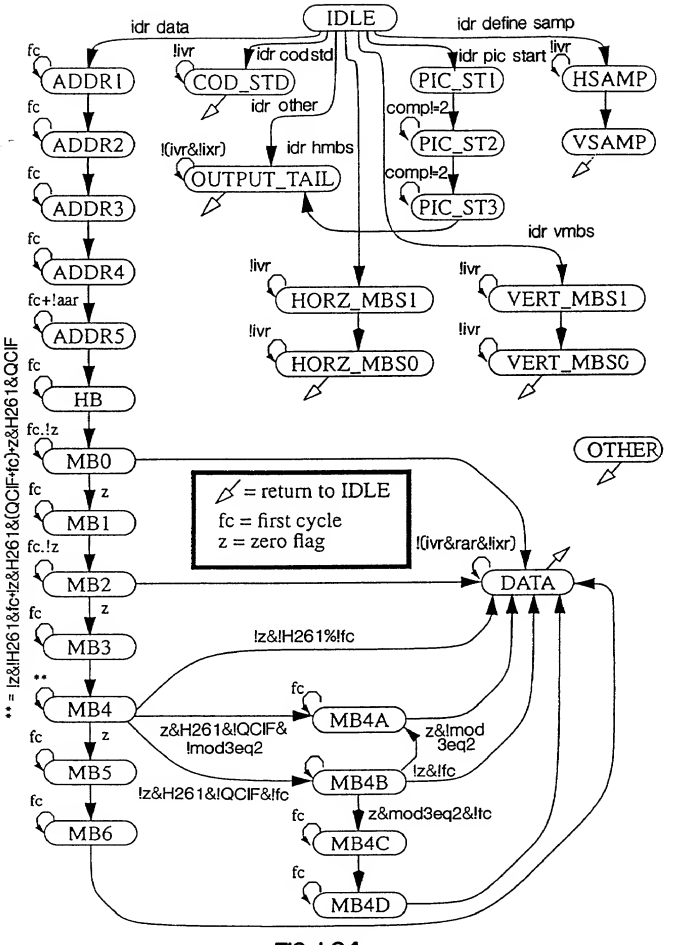

IDLE
idr data
idr cod std
idr other
idr hmbs
idr vmbs
idr pic start
idr define samp
ADDR1
ADDR2
ADDR3
ADDR4
ADDR5
HB
MB0
MB1
MB2
MB3
MB4
MB5
MB6
COD_STD
OUTPUT_TAIL
HORZ_MBS1
HORZ_MBS0
PIC_ST1
PIC_ST2
PIC_ST3
HSAMP
VSAMP
VERT_MBS1
VERT_MBS0
OTHER
DATA
MB4A
MB4B
MB4C
MB4D
fc
!ivr
compl=2
fc+!aar
fc.!z
z
!(ivr&lixr)
= return to IDLE
fc = first cycle
z = zero flag
!(ivr&rar&!ixr)
!z&!H261%!fc
z&H261&!QCIF&
!mod3eq2
z&!mod
3eq2
!z&!fc
!z&H261&!QCIF&!fc
z&mod3eq2&!fc
** = !z&!H261&!f+!z&H261&(QCIF+fc)+z&H261&QCIF

FIG. 164

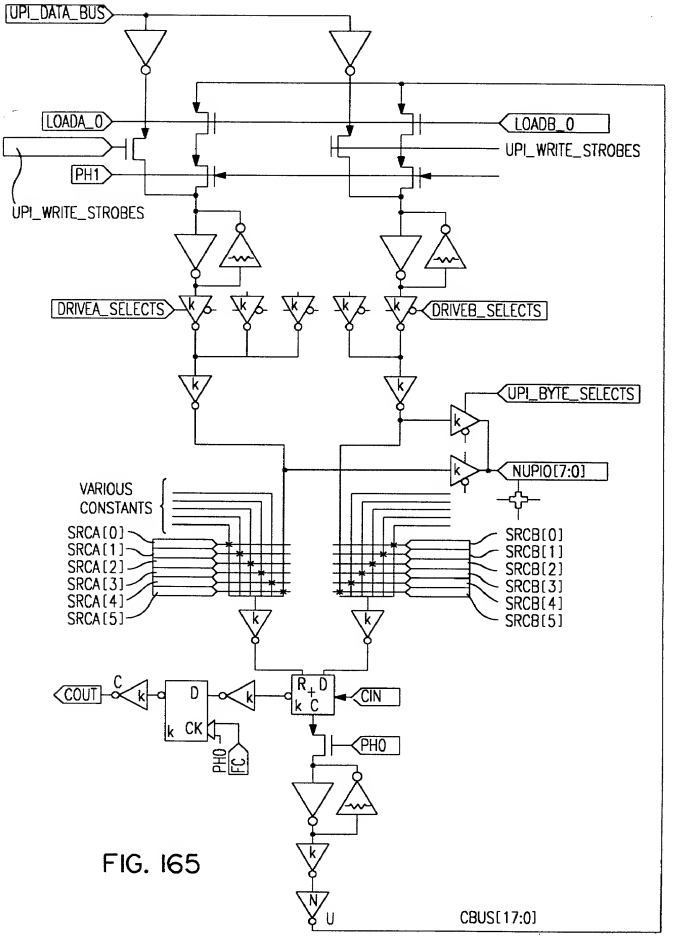

UPI_DATA_BUS
LOADA_0
LOADB_0
UPI_WRITE_STROBES
PH1
UPI_WRITE_STROBES
DRIVEA_SELECTS
DRIVEB_SELECTS
UPI_BYTE_SELECTS
NUPIO[7:0]
VARIOUS
CONSTANTS
SRCA[0]
SRCA[1]
SRCA[2]
SRCA[3]
SRCA[4]
SRCA[5]
SRCB[0]
SRCB[1]
SRCB[2]
SRCB[3]
SRCB[4]
SRCB[5]
COUT
C
D
CK
PH0
FC
R
D
C
CIN
PH0
N
U
CBUS[17:0]

FIG. 165

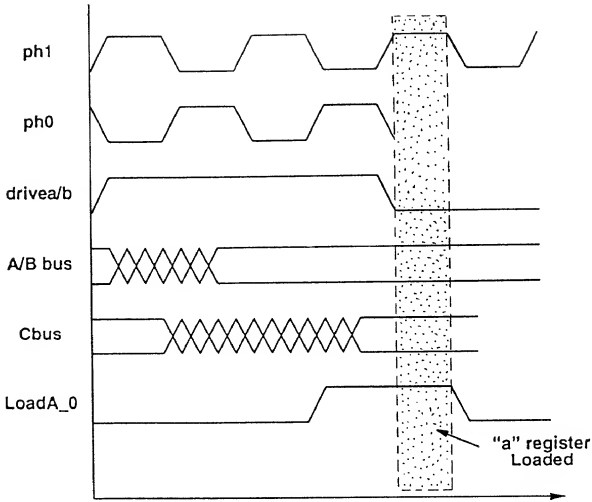


FIG. I 66

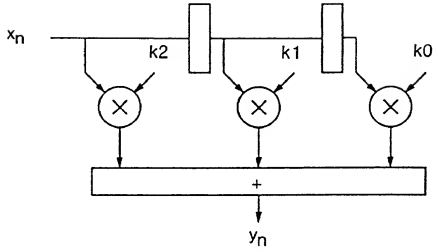


FIG. 167

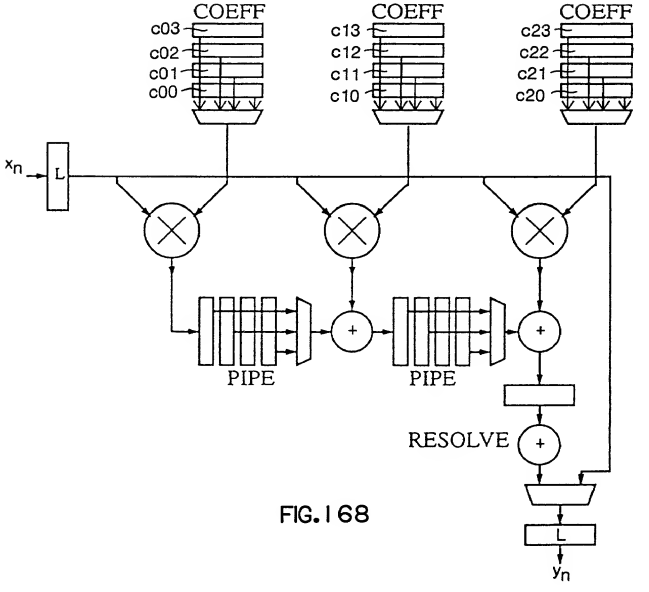


FIG. 168

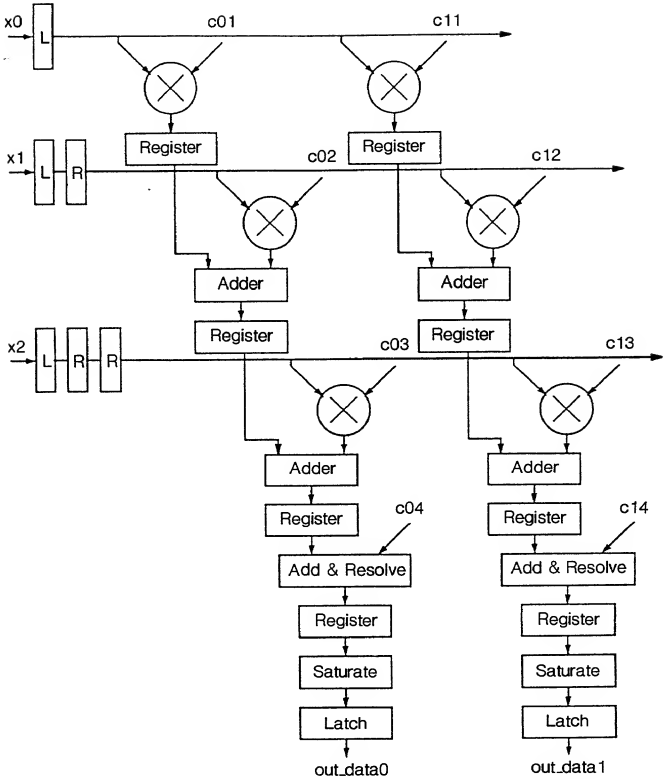

x0
L
c01
c11
Register
Register
x1
L
R
c02
c12
Adder
Adder
Register
Register
x2
L
R
R
c03
c13
Adder
Adder
Register
Register
c04
c14
Add & Resolve
Add & Resolve
Register
Register
Saturate
Saturate
Latch
Latch
out_data0
out_data1

FIG. 169